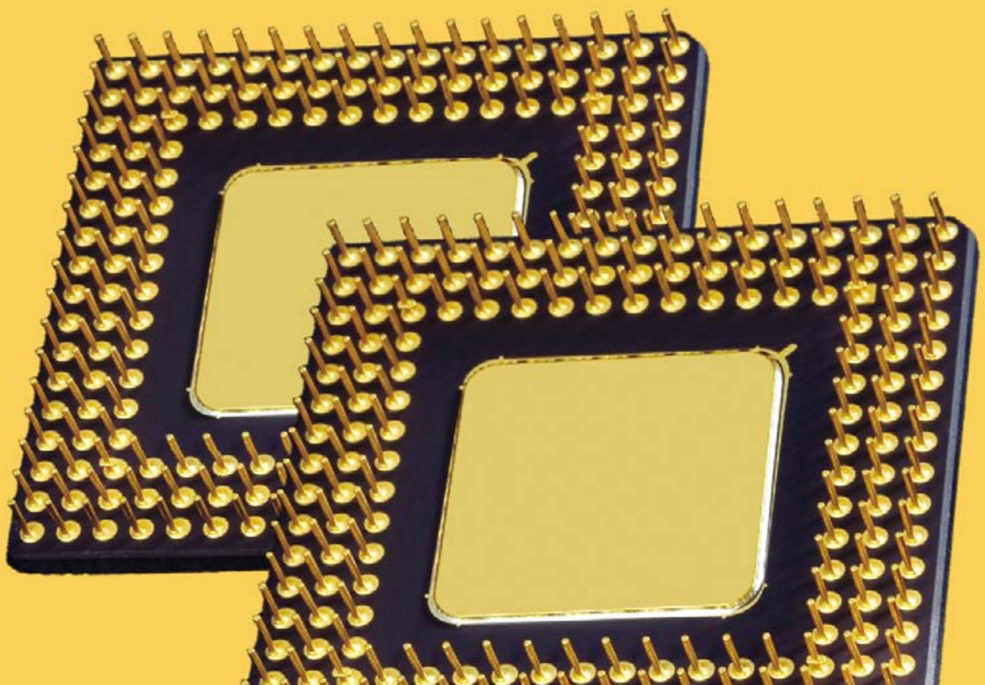
NEW AGE

SECOND EDITION

UNDERSTANDING 8085/8086 MICROPROCESSORS and PERIPHERAL ICs

Through Questions and Answers



S. K. Sen

NEW AGE INTERNATIONAL PUBLISHERS

UNDERSTANDING 8085/8086
MICROPROCESSORS
and PERIPHERAL ICs
Through Questions and Answers

This page intentionally left blank

UNDERSTANDING 8085/8086
MICROPROCESSORS
and PERIPHERAL ICs
Through Questions and Answers
(SECOND EDITION)

S. K. Sen
B.Tech., M.Tech., Ph.D.
Professor and Head
Instrumentation Engineering Division
Department of Applied Physics
University of Calcutta, Kolkata

PUBLISHING FOR ONE WORLD

NEW AGE INTERNATIONAL (P) LIMITED, PUBLISHERS
New Delhi • Bangalore • Chennai • Cochin • Guwahati • Hyderabad
Jalandhar • Kolkata • Lucknow • Mumbai • Ranchi
Visit us at **www.newagepublishers.com**

Published by New Age International (P) Ltd., Publishers

ISBN (13) : 978-81-224-2974-9

PUBLISHING FOR ONE WORLD

NEW AGE INTERNATIONAL (P) LIMITED, PUBLISHERS
4835/24, Ansari Road, Daryaganj, New Delhi - 110002
Visit us at **www.newagepublishers.com**

Preface to the Second Edition

It is heartening to note that the text is going to the second edition. Inadvertently, numerous typographical errors crept into the first edition - which are being taken care of in this edition.

Requests from many faculty as well as students have led the author to include 8086 Microprocessor and its peripheral ICs in an elaborate fashion to the existing book.

The author is open to suggestions for improvements of the book.

S.K. Sen

This page intentionally left blank

Preface to the First Edition

Since the advent of microprocessors in the 70s, a good many books have been written covering different aspects of microprocessors and microcomputers. A knowledge and exposure to microprocessors is a must for practising engineers and scientists in the fields of Electrical, Electronics, Instrumentation and Software Engineering.

This book, written in a problem-solution form, is primarily intended for undergraduate students of Engineering who have already gone through the course on microprocessors, but is at sea in finding out the right answers to the numerous questions that confront them. In my view, this book is an answer to their long list of queries.

Amongst the microprocessors available, 8085 has been taken up for discussion because it is a time-tested 8 bit microprocessor and still very much in use today. Smaller bit microprocessors have not much relevance and larger bit microprocessors are merely an extension of this basic microprocessor. Thus, a thorough understanding of 8085 microprocessor is central and is a gateway to the more powerful range of microprocessors in use today.

The book begins with a discussion on microprocessor, microcomputer and associated languages in Chapter 1 followed by a detailed discussion on 8085 microprocessor in Chapter 2 and instruction types and timing diagrams in Chapter 3. Interrupt details of 8085 are taken up for discussion in Chapter 4 while programming techniques are discussed in Chapter 5. Chapter 6 discusses stack and subroutines, while data transfer technique and interfacing are taken up for discussion in Chapter 7. A detailed discussion on memory is included in Chapter 8. Different peripherals like 8255, 8155/8156, 8355/8755, 8279, 8259, 8257, 8253, 8254 and 8251 are discussed in Chapter 9, while the last and concluding Chapter discusses bus standards RS-232C and IEEE-488.

This comprehensive book on microprocessor and peripheral ICs will cater to the needs of engineering students, and for students appearing and aspiring for GATE, IETE, AMIE and other all India level examinations conducted by various public sector undertakings.

Suggestions for improvements in any form will be highly appreciated by the author.

S.K. Sen

This page intentionally left blank

Contents

1

Microprocessor, Microcomputer and Associated Languages

1. On which model is based the basic architecture of a digital computer?

Ans. The basic architecture of a digital computer is based on Von Neumann model.

2. What is meant by distributed processing?

Ans. Distributed processing involves the use of several microprocessors in a single computer system. For example, for such a system, the first microprocessor may control keyboard activities, the second controls storage devices like disk drives, the third controls input/output operations, while the fourth may act as the main system processor.

3. When was the first microprocessor developed?

Ans. The first microprocessor was developed by BUSICOM of Japan and INTEL of USA in the year 1971.

4. What is a microprocessor?

Ans. A microprocessor may be thought of as a silicon chip around which a microcomputer is built.

5. What is the technology used in microprocessors?

Ans. NMOS technology is used in microprocessors.

6. What are the three main units of a digital computer?

Ans. The three main units of a digital computer are: the central processing unit (CPU), the memory unit and the input/output devices.

7. How does the microprocessor communicate with the memory and input/output devices?

Ans. The microprocessor communicates with the memory and the Input/Output devices via the three buses, viz., data bus, address bus and control bus.

8. What are the different jobs that the CPU is expected to do at any given point of time?

Ans. The CPU may perform a memory read or write operation, an I/O read or write operation or an internal activity.

9. What is a mnemonic?

Ans. It is very difficult to understand a program if it is written in either binary or hex code. Thus the manufacturers have devised a symbolic code for each instruction, called a mnemonic.

Examples of mnemonics are:
INR A, ADD M, etc.

10. What is machine language programming?

Ans. Programming a computer by utilising hex or binary code is known as machine language programming.

11. What is meant by assembly language programming?

Ans. Programming a microcomputer by writing mnemonics is known as assembly language programming.

12. What are meant by low level and high level languages?

Ans. Programming languages that are machine dependent are called low level languages. For example, assembly language is a low level language.

On the other hand, programming languages that are machine independent are called high level languages. Examples are BASIC, FORTRAN, C, ALGOL, COBOL, etc.

13. What is meant by 'word length' of a computer?

Ans. The number of bits that a computer recognises and can process at a time is known as its 'word length'.

14. What is meant by instruction?

Ans. An instruction is a command which asks the microprocessor to perform a specific task or job.

15. How many different instructions µP 8085 has? What is an instruction set?

Ans. 8085 microprocessor has a total of 74 different instructions for performing different operations or tasks.

The entire different instructions that a particular microprocessor can handle is called its instruction set.

16. What an instruction consists of?

Ans. An instruction consists of an operation code (called 'opcode') and the address of the data (called 'operand'), on which the opcode operates.

Operation code (or opcode)	Address of data (or operand)
Field 1	Field 2

17. Give one example each of the different types of instructions.

Ans. Instructions can be of (*i*) direct (*ii*) immediate (*iii*) implicit type. Examples of each type follows:

(*i*) direct type : LDA 4000
(*ii*) immediate type : MVIA, 1F
(*iii*) implicit type : ADD C

18. What language a microprocessor understands?

Ans. Microprocessor understands only binary language.

19. How the mnemonics written in assembly language are translated into binary?

Ans. The translation from assembly language (i.e., mnemonics) into binary is done either manually (known as hand (or manual) assembly) or by a program called an assembler.

Thus an assembler can be thought of as a program which translates the mnemonics entered by an ASCII keyboard into its equivalent binary code, which is the only one understood by a microprocessor.

20. How an assembler translates programs written in mnemonic form to binary?

Ans. An assembler has a 'translation dictionary', which is stored in its memory. Mnemonics entered via keyboard is compared with this dictionary, which then retrieves its binary equivalent from the same place (dictionary).

21. What are the types of mnemonics possible?

Ans. Mnemonics can be of two types—alphabetical or alphanumerical.

Example of first type is ADD D whereas, MVI A, 0F is an example of the second category.

22. What are source codes and object codes?

Ans. High level languages (like, BASIC, COBOL, ALGOL, etc.) are called source codes, whereas binary language is called object code.

23. How are high level languages converted into binary?

Ans. The high level languages are converted into their corresponding binary by means of another program, called either a compiler or an interpreter.

Compilers are generally used for FORTAN, PASCAL, COBOL, etc. whereas interpreters are generally used in BASIC. M-BASIC is a common example of an interpreter for BASIC language.

24. What is a 'statement'?

Ans. Instructions written in high level languages are known as statements.

25. Write down the difference between a compiler and an interpreter.

Ans. Difference between a compiler and an interpreter lies in the generation of the machine code or object code.

A compiler reads the entire program first and then generates the corresponding object code.

Whereas, an interpreter reads an instruction at a time, produces the corresponding object code and executes the same before it starts reading the next instruction. A program from a compiler runs some 5 to 25 times faster than a program from an interpreter.

26. Differentiate between a compiler/interpreter and an assembler.

Ans.

S.No.	Compiler/Interpreter	Assembler
1.	Debugging easier	Debugging relatively tougher
2.	Less efficient	More efficient
3.	Requires large memory space	Requires less memory space

As an example, for gadget controls and traffic signals, assembly language programming is done because programs in such cases are compact and not lengthy. But for cases where large program lengths are a must, compilers/interpreters are used. For such a case, although a large memory space is required, the advantage lies in very quick debugging.

27. What are the names given to instructions written in high and low level languages?

Ans. The instructions written in high level languages are known as 'statements' whereas those written in low level languages are called 'mnemonics'.

28. What is another name of a microprocessor?

Ans. A microprocessor is also called the CPU, the central processing unit.

29. What is a microcomputer?

Ans. A microcomputer is a system which is capable of processing a stream of input informations and will generate a stream of output informations taking the help of a program which is stored in the memory of the system.

30. What are the jobs that a microcomputer is capable of doing? How does it do the jobs?

Ans. A microcomputer is capable of doing the following jobs:

(*a*) receiving input (in the form of data or instruction).

(*b*) performing computations, both arithmetic and logical.

(*c*) storing data and instructions.

(*d*) displaying the results of any computations.

(*e*) controlling all the devices that perform the above mentioned four tasks, either directly or indirectly.

The first task is done by an input device, whereas the second job is done by the arithmetic logic unit (ALU). The third task is done by the memory unit while an output device does the fourth job.

The fifth job is executed by the timing and control (T&C) unit.

31. What is a bus?

Ans. A bus is a bunch of wires through which data or address or control signals flow.

32. What are the different buses and what jobs they do in a microprocessor?

Ans. The different buses in a microprocessor are the data bus (DB), address bus (AB) and the control bus (CB). Data flow through the DB, while address comes out of the AB and CB controls the activities of the microcomputer system at any instant of time.

33. Why are the different buses buffered?

Ans. A buffer enhances the current or power driving capability of a pin of an IC.

34. In how many ways computers are divided?

Ans. Computers are divided into three categories as per the superiority and number of microprocessors used. These are:

- Micro computers
- Mini computers
- Mainframe computers

35. Distinguish between the three types of computers.

Ans. A microcomputer is a small computer containing only a single central processing unit (CPU). Their word length varies between 8 and 32 bits and used in small industrial and process control systems. The storage capacity and speed requirements of microcomputers are moderate.

Mini computers are having more storage capacity and more speed than micro computers. Mini computers are used in research, data processing, scientific calculations etc.

Mainframe computers are designed to work at very high speed and they have very high storage capacity. Their word length is typically 64-bits. These are used for research, data processing, graphic applications, etc.

36. In how many ways computer softwares are categorised?

Ans. Computer softwares are divided into two broad categories—system software and user software.

37. Explain the two types of softwares.

Ans. System software is a collection of programs used for creation, preparation and execution of other programs. Editors, assemblers, loaders, linkers, compilers, interpreters, debuggers and OS (operating system) are included in system software.

User software are softwares developed by various users.

38. Draw the software hierarchy of a microcomputer system.

Ans. The software hierarchy is shown below:

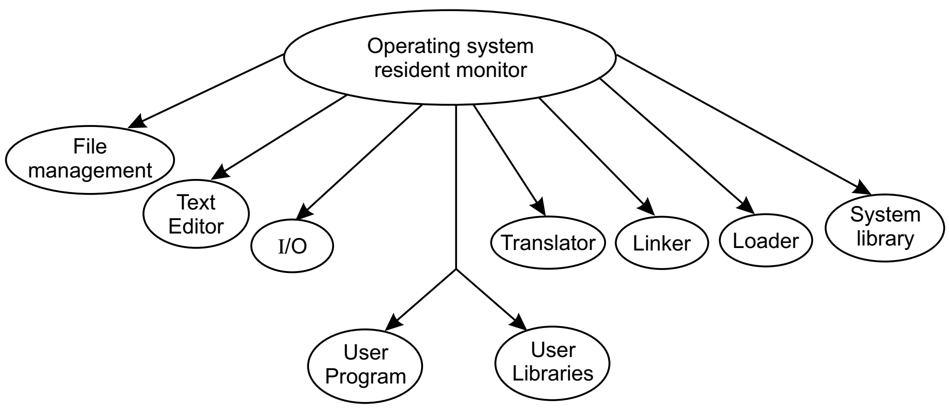


Fig. 1.1: Software hierarchy of a microcomputer system

39. What is an editor?

Ans. An editor is a program and is used for creation and modification of source programs or text. The editor program includes commands which can delete, insert or change lines/characters.

An editor stores in ASCII form in successive RAM locations of the typed letters/numbers. This then can be saved by the SAVE command on a floppy or hard disk. The editor can load the program on RAM later for corrections/alterations, if necessary.

40. What is an OS (operating system) and what are its functions?

Ans. An operating system provides an interface between the end user and the machine. It performs the function of resource management. By 'resource' is meant a microprocessor, memory or an I/O device.

An OS is a collection of a set of programs that manages machine functions under varied conditions.

The different functions performed by the OS include an efficient or effective way of sharing memory, I/Os and the CPU amongst different users. Today, operating systems are DOS, UNIX, WINDOWS, etc.

41. What are the different types of assemblers used?

Ans. The different assemblers used are: (*a*) Macro Assembler, (*b*) Cross Assembler, (*c*) Meta Assembler.

42. What is a linker?

Ans. A linker is a program that links several small object files to produce one large object file.

A large program is usually divided into several small programs. They are written separately, tested and debugged. The large program is then produced by linking these debugged programmes.

A linker has a link file and a link map. The former contains the binary codes of all the modules which have been combined while the link map stores in it the addresses of all the link files. A linker assigns relative addressing instead of absolute addressing which helps in putting a linker program anywhere in the memory.

43. What is a locator?

Ans. A locator is a program which is used to assign specific address when object code is to be loaded into the memory.

44. What are the different assembly languages used for 8085 microprocessor?

Ans. Two different assembly languages are used for 8085 microprocessor—one is used by the manufacturer INTEL and the other defined in IUL 77.

45. What is a coprocessor?

Ans. A coprocessor or an arithmetic coprocessor, as it is so called, performs operations like exponentiation, trigonometric functions, etc. To implement these functions in CPU hardware is a costly one and the software implementation of the above slows down the processor. A coprocessor overcomes these difficulties.

The coprocessor has its own instruction set. The CPU and the coprocessor execute their respective instructions from the same program.

The instructions belonging to the coprocessor are fetched and decoded by the CPU but executed by the coprocessor.

46. Draw a typical coprocessor configuration and discuss the same.

Ans. A typical coprocessor configuration looks like as follows:

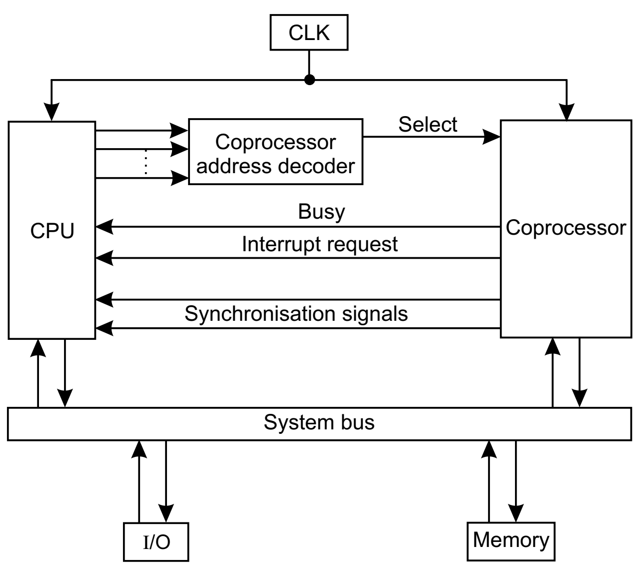


Fig. 1.2: A typical coprocessor configuration

The CPU and the coprocessor share the same clock, bus control logic and also share the entire memory and the I/O subsystem. In normal configuration, CPU is the master and the coprocessor is the slave.

The CPU treats the coprocessor as an external memory in the sense that the CPU can read from/write into the coprocessor registers in a manner alike that of an external memory. The CPU and the coprocessor executes their instructions independent of each other. When the coprocessor executes instructions, it indicates the CPU about its executions via the 'BUSY' signal.

47. What is a coprocessor trap?

Ans. When a coprocessor is not connected in the system and the functions of the coprocessor is done by the CPU by writing coprocessor instructions from a predetermined location in the form of a software routine and invoked by interrupt, it is known as 'coprocessor trap'.

48. Explain the three fields contained in a coprocessor instruction.

Ans. There are three fields in a coprocessor instruction—these are code, address and opcode.

The code field distinguishes between the coprocessor and CPU instructions. The address field indicates the address of a particular coprocessor. This is required when several coprocessors are used in the system. The opcode field indicates the type of operation to be executed by the processor.

49. What is a debugger?

Ans. A debugger is a program that debugs an object code program by placing it into the system memory and subsequently executing the same.

50. How does a debugger help in debugging a program?

Ans. A debugger is a software tool which isolates the problems/drawbacks in a programmer's program.

A debugger helps in debugging a program in the following ways:

(*a*) A debugger helps in checking the contents of memory locations and various registers and also alter the same if required and rerun the program to check the correctness of the modified program.

(*b*) Program execution can be stopped after each instruction–hence step by step checking is possible with a debugger.

(*c*) A debugger can set a breakpoint at any place in a program.

A program then can run up to the breakpoint address and not beyond. Thus any fault in the program up to the breakpoint address can be checked by having a look at the various registers and memory contents. If no fault occurs the breakpoint is set at a latter address in the program and the debugger can again be rerun to check for the correctness of the program. This way the whole program can be corrected by judiciously inserting breakpoints in a program.

51. What is meant by the term 'word'?

Ans. A collection of bits is called a word. A word does not have a fixed number of bits—unlike the case of byte (= 8 byte) or a nibble (= 4 bytes).

For different microprocessors, the word length varies. For example, for 8-bit microprocessors, the word length is 8-bits while that for 32-bit microprocessors, the word length is 32-bits.

A word is expressed usually in multiples of 2, but no value is excluded. That is, we can have a 19-bit word or 37-bit word, etc.

52. What is meant by the term 'long word'?

Ans. The term 'long word' means a group of bits twice the normal length of the microprocessor. Hence, for a 16-bits microprocessor like 8086, a long word means a group consisting of 32-bits.

53. Distinguish between KB, MB, GB, TB and PB.

Ans. These stand for Kilobyte, Megabyte, Gigabyte, Terabyte and Petabyte respectively.

1 KB	=	1024 bytes	=	2^{10} bytes
1 MB	=	1 Kilo KB	=	2^{20} bytes
1 GB	=	1 Kilo MB	=	2^{30} bytes
1 TB	=	1 Kilo GB	=	2^{40} bytes
1 PB	=	1 Kilo TB	=	2^{50} bytes

54. Compare signed magnitude number and complementary numbers.

Ans. The comparison is made on the basis of an 8-bit word.

In signed magnitude number system, MSB represents the sign of the number with 1 and 0 representing minus and plus, respectively. This number system has the following drawbacks:

Using the MSB for sign bit leaves 7-bits for data which can range from 0 to 127_{10} (= 1111111_2) while 8-bits can manage numbers from 0 to 255_{10} (=$1111\ 1111_2$).

Addition of two signed numbers does not give the correct result always. An example follows:

Two decimal numbers $+127_{10}$ and $+8_{10}$ are to be added via signed number system.

$+127_{10}$	:	The Signed no. is	=	0111 1111
$+8_{10}$	:	The Signed no. is	=	0000 1000
			(+)	1000 0111

This represents the sign of the number. Here it is negative. (first bit)

This represents the magnitude of the number. Here it is 07. (remaining bits)

Thus the result of addition of the two numbers is -07_{10}, but it should have been $+135_{10}$.

Use of complementary numbers (usually 2's complement of a binary number is used) gives rise to the following advantages:

- For an 8-bit data, the data values can vary from 0 to 255_{10} (= $1111\ 1111_2$)
- Addition and subtraction can be implemented with almost same circuitry.

55. What is meant by normalising a number?

Ans. When a number 4751 is converted to 4.751×10^3, the latter is called the normalised version of the former.

Likewise, in binary, a number 1011 1101 can be normalised to 1.0111101×2^8. Another binary number 0.0001101 is normalised to 1.101×2^{-4}.

If a given binary number is in a form 1.0101101, then it is already in the normalised form.

56. Name the different parts of a normalised number.

Ans. Let a binary number is 1.01101×10^{11}. Then

$$1.01101 \times 10^{11}$$

Mantissa (1.01101) Radix (10) Exponent (11)

Hence, a normalised binary number consists of mantissa, radix and exponent.

57. Discuss the utility of floating point numbers.

Ans. Floating point numbers are useful when the system has either to handle extremely large numbers (as in Astronomy) or very small ones (as in Nuclear Physics). For floating point numbers, the binary number starts with a 1, followed by the binary point. An 8-bit binary mantissa can assume extreme values as 1.0000000 to 1.1111111.

While storing such floating point numbers, the mandatory 1 followed by the binary point can safely be ignored—thereby saving previous storage space. It is assumed that the 1 and the binary point are just present. Thus the above can now be stored as 0000000 to 1111111.

For floating point numbers, a 32-bit storage area is used. Many variants are there to fill in these 32-bits, but the following figure is the one that is most sought after.

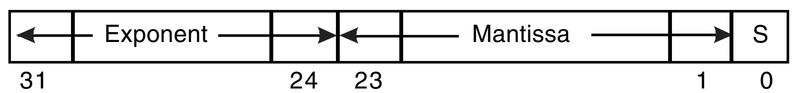


Fig. 1.3: Floating point numbers—32-bit storage area

Bit 0 is the sign bit for the mantissa. Bits 1 to 23 hold the mantissa while bits 24 to 31 hold the exponent of the number. The eight bits (bits 24 to 31) are used to hold the numbers from –127 to +128 in 2's complement form or Excess –127 form. In Excess –127 notation, 127 is added to the original exponent value (in decimal)—this is done to get rid of the negative exponent values. Thus this modified exponent value (positive) is stored in bits 24 to 31 after it is converted into binary.

58. Discuss speed, size and accuracy while handling floating point numbers.

Ans. Floating point operations that can be carried out per second is a measure of the speed—abbreviated as FLOPS.

Accuracy is dependent on the number of mantissa bits. More number of mantissa bits result in more accuracy. Bits 1 to 23, meant for mantissa, can have a maximum binary value of 1.1111 1111 1111 1111 1111 111. The 1's, starting from the binary point on the right, have their decimal values +0.5, +0.25, +0.125, +0.0625, etc. Thus, when the 24-bits are converted into decimal, it will approach a value of 2 — but will never equal it. If more number of bits are accommodated in mantissa (i.e., mantissa will now have more than the normal 24-bits), then the converted decimal equivalent will be having a value which will be even more closer to 2. Thus increasing number of mantissa bits result in more *accuracy*.

Size is dependent on the number of exponent bits. More number of exponent bits results in greater size of the number that can be handled. Bits 24 to 31, meant for exponent, has eight bits ranging from –127 to +128. Thus the maximum number that can be handled is $1 \times 2^{128} = 3.4 \times 10^{38}$ roughly. This value would have been more if in case more number of bits are accommodated into exponent bits. Thus increasing number of exponent bits results in greater *size* of the number that can be handled. In conclusion, when more accuracy is demanded, number of mantissa bits are to be increased (instead of the normal 24-bits) and in cases of higher size of data handling, number of exponent bits are to be increased (instead of the normal 8-bits) within the 32-bit storage space.

59. Explain single and double precision.

Ans. Single and double precision mean 32-bits and 64-bits storage areas respectively. The extra storage space made available from single to double precision—utilised for

accommodating more number of mantissa bits—would result in much more accuracy from the system.

60. List the different generation languages.

Ans. These are first, second, third and fourth generation languages. Machine code belongs to first generation, while Assembly Language belongs to second generation language. First and second generation languages are known as low level languages. Third generation languages include FORTRAN, BASIC, COBOL, PASCAL, C, C^{++}, JAVA while fourth generation languages are LISP, APL, PROLOG, etc.

Fig. 1.4 shows pictorially different generation languages.

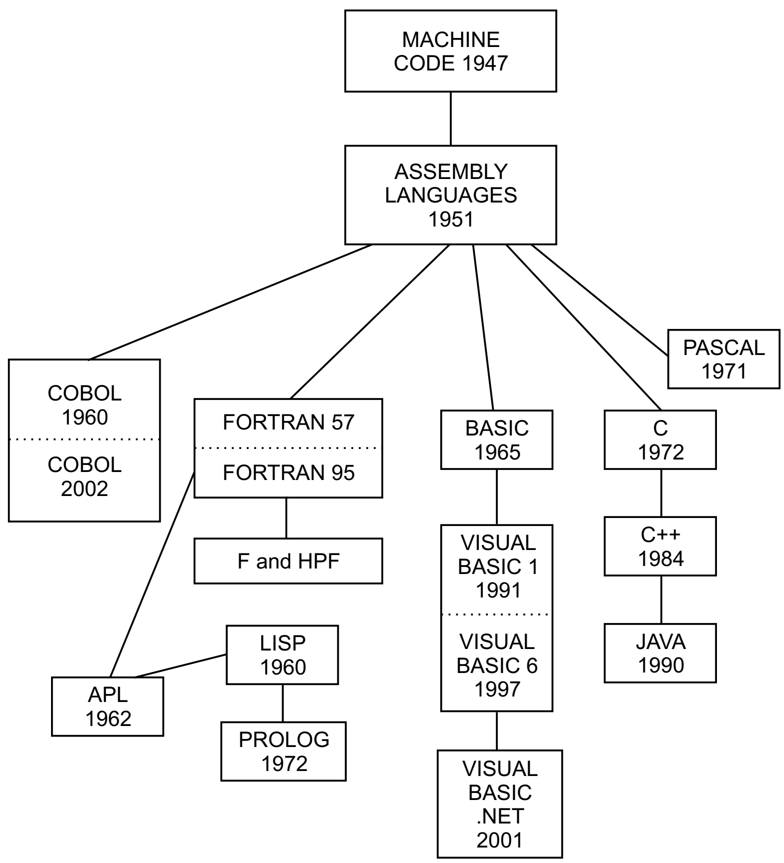


Fig. 1.4: The different languages

2

The 8085 Microprocessor

1. Draw the pin configuration and functional pin diagram of μP 8085.

Ans. The pin configuration and functional pin diagram of μP 8085 are shown below:

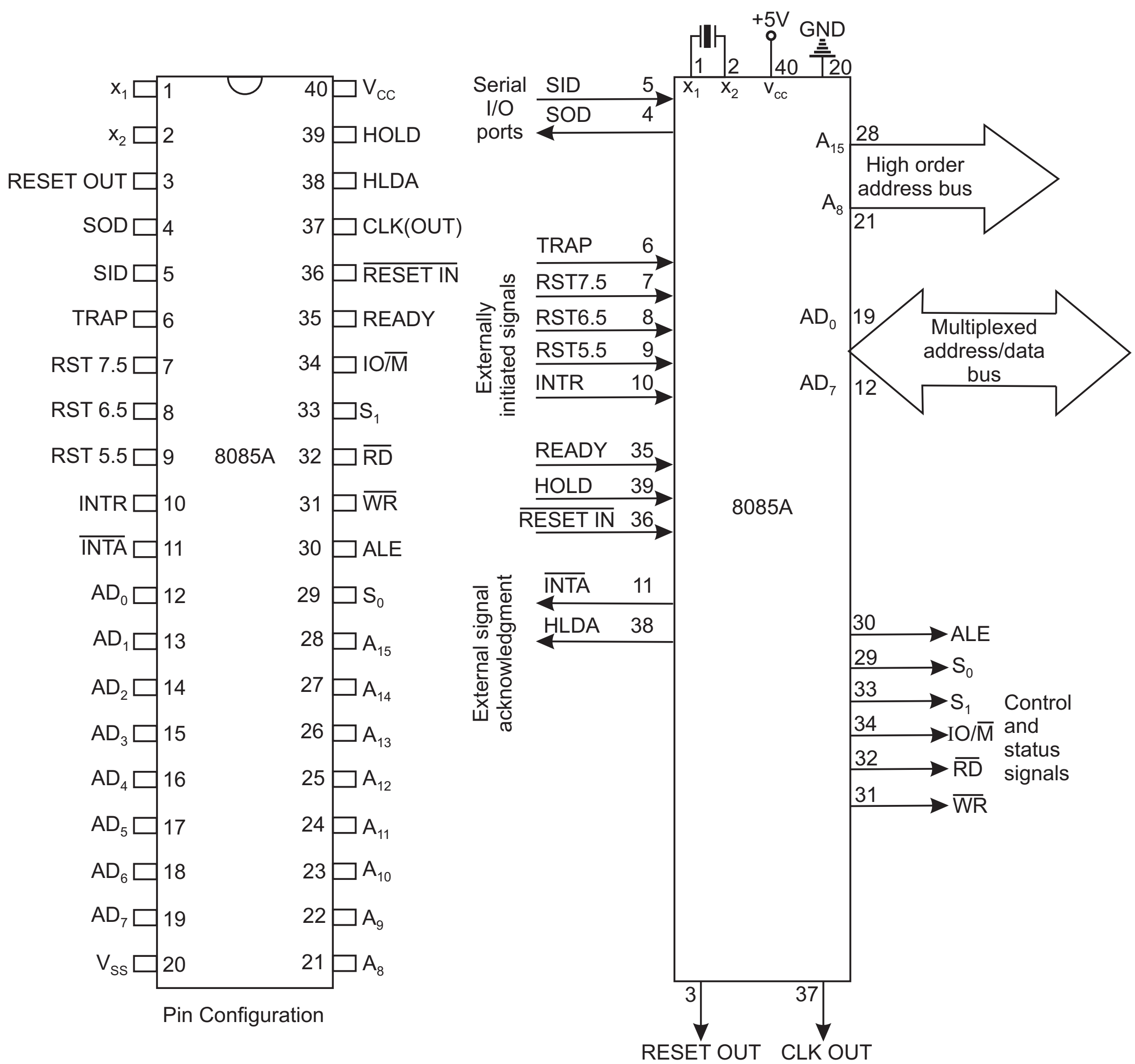


Fig. 2.1: Functional pin diagram

2. In how many groups can the signals of 8085 be classified?

Ans. The signals of 8085 can be classified into seven groups according to their functions. These are:

(1) Power supply and frequency signals (2) Data and Address buses (3) Control bus (4) Interrupt signals (5) Serial I/O signals (6) DMA signals (7) Reset signals.

3. Draw the architecture of 8085 and mention its various functional blocks.

Ans. The architecture of 8085 is shown below:

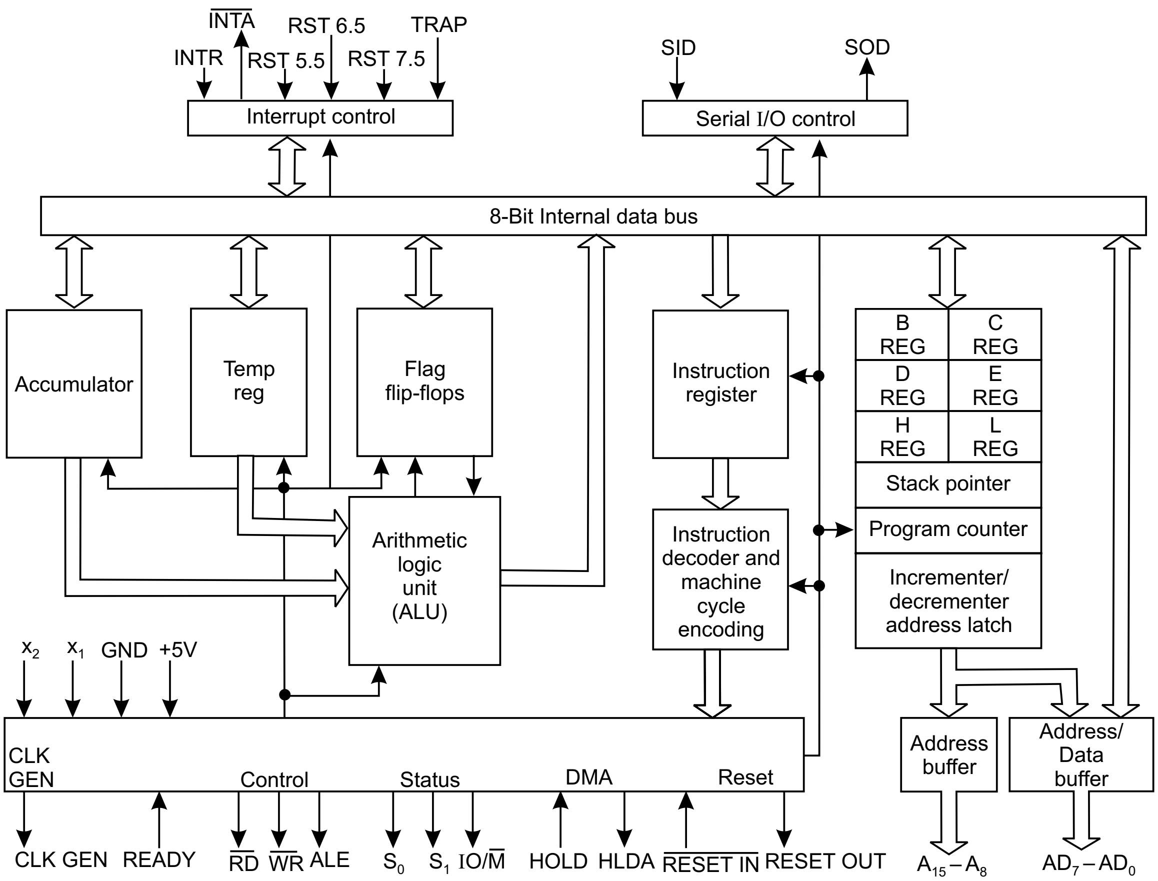


Fig. 2.2: Architecture of 8085

The various functional blocks of 8085 are as follows:

- Registers
- Arithmetic logic unit
- Address buffer
- Incrementer/decrementer address latch
- Interrupt control
- Serial I/O control
- Timing and control circuitry
- Instructions decoder and machine cycle encoder.

4. What is the technology used in the manufacture of 8085?

Ans. It is an NMOS device having around 6200 transistors contained in a 40 pin DIP package.

5. What is meant by the statement that 8085 is a 8-bit microprocessor?

Ans. A microprocessor which has n data lines is called an n-bit microprocessor i.e., the width of the data bus determines the size of the microprocessor. Hence, an 8-bit microprocessor like 8085 can handle 8-bits of data at a time.

6. What is the operating frequency of 8085?

Ans. 8085 operates at a frequency of 3 MHz, and the minimum frequency of operation is 500 kHz.

The version 8085 A-2 operates at a maximum frequency of 5 MHz.

7. Draw the block diagram of the built-in clock generator of 8085.

Ans. The built-in clock generator of 8085, in block schematic, is shown below:

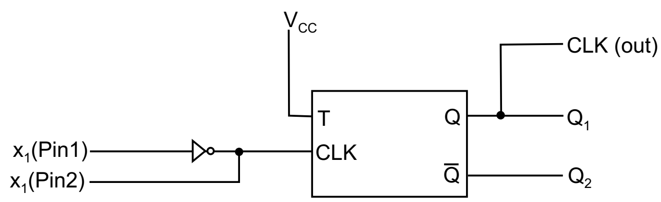


Fig. 2.3: Block diagram of built-in clock generator

The internal built-in clock generator, LC or RC tuned circuits, piezo-electric crystal or external clock source acts as an input to generate the clock. The T F/F, shown in Fig. 2.3 divides the input frequency by 2. Thus the output frequency of 8085 (obtained from pin 37) is half the input frequency.

8. What is the purpose of CLK signal of 8085?

Ans. The CLK (out) signal obtained from pin 37 of 8085 is used for synchronizing external devices.

9. Draw the different clock circuits which can be connected to pins 1 and 2 of 8085.

Ans. The different external clock circuits which can be connected to pins 1 and 2 of 8085 are shown below in Fig. 2.4:

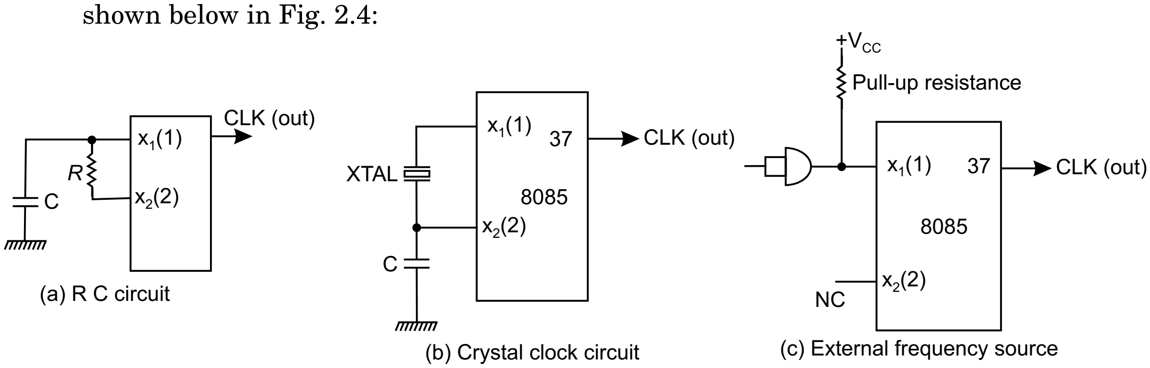


Fig. 2.4: The different external clock circuits

The output frequency obtained from pin 37 of Fig. 2.4(b) is more stable than the RC circuit of Fig. 2.4(a).

10. What are the widths of data bus (DB) and address bus (AB) of 8085?

Ans. The width of DB and AB of 8085 are 8-bits (1 byte) and 16-bits (2 bytes) respectively.

11. What is the distinguishing feature of DB and AB?

Ans. While the data bus is bidirectional in nature, the address bus is unidirectional.

Since the μP can input or output data from within it, hence DB is bidirectional. Again the microprocessor addresses/communicates with peripheral ICs through the address bus, hence it is unidirectional, the address comes out via the AB of μP.

12. The address capability of 8085 is 64 KB. Explain.

Ans. Microprocessor 8085 communicates via its address bus of 2-bytes width – the lower byte $AD_0 – AD_7$ (pins 12-19) and upper byte $D_8 – D_{15}$ (pins 21–28). Thus it can address a maximum of 2^{16} different address locations. Again each address (memory location) can hold 1 byte of data/instruction. Hence the maximum address capability of 8085 is

$$= 2^{16} \times 1 \text{ Byte}$$

$$= 65,536 \times 1 \text{ Byte}$$

$$= 64 \text{ KB} \quad (\text{where } 1 \text{ K} = 1024 \text{ bytes})$$

13. Does 8085 have serial I/O control?

Ans. 8085 has serial I/O control via its SOD and SID pins (pins 4 and 5) which allows it to communicate serially with external devices.

14. How many instructions 8085 can support?

Ans. 8085 supports 74 different instructions.

15. Mention the addressing modes of 8085.

Ans. 8085 has the following addressing modes: Immediate, Register, Direct, Indirect and Implied.

16. What jobs ALU of 8085 can perform?

Ans. The Arithmetic Logic Unit (ALU) of 8085 can perform the following jobs:

- 8-bit binary addition with or without carry.
- 16-bit binary addition.
- 2-digit BCD addition.
- 8-bit binary subtraction with or without borrow.
- 8-bit logical OR, AND, EXOR, complement (NOT function).
- bit shift operation.

17. How many hardware interrupts 8085 supports?

Ans. It supports five (5) hardware interrupts—TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.

18. How many I/O ports can 8085 access?

Ans. It provides 8-bit I/O addresses. Thus it can access $2^8 = 256$ I/O ports.

19. Why the lower byte address bus ($A_0 - A_7$) and data bus ($D_0 - D_7$) are multiplexed?

Ans. This is done to reduce the number of pins of 8085, which otherwise would have been a 48 pin chip. But because of multiplexing, external hardware is required to demultiplex the lower byte address cum data bus.

20. List the various registers of 8085.

Ans. The various registers of 8085, their respective quantities and capacities are tabulated below:

Table 2.1: List of Various Registers in 8085

S. No.	Name of the Register	Quantity	Capacity
1.	Accumulator (or) Register A	1	8-bit
2.	Temporary register	1	8-bit
3.	General purpose registers (B, C, D, E, H and L)	6	8-bit each
4.	Stack pointer (SP)	1	16-bit
5.	Program counter (PC)	1	16-bit
6.	Instruction register	1	8-bit
7.	Incrementer/Decrementer address latch	1	16-bit
8.	Status flags register	1	8-bit

21. Describe the accumulator register of 8085.

Ans. This 8-bit register is the most important one amongst all the registers of 8085. Any data input/output to/from the microprocessor takes place via the accumulator (register). It is generally used for temporary storage of data and for the placement of final result of arithmetic/logical operations.

Accumulator (ACC or A) register is extensively used for arithmetic, logical, store and rotate operations.

22. What are the temporary registers of 8085?

Ans. The temporary registers of 8085 are temporary data register and W and Z registers. These registers are not available to the programmer, but 8085 uses them internally to hold temporary data during execution of some instructions.

23. Describe W and Z registers of 8085.

Ans. W and Z are two 8-bit temporary registers, used to hold 8-bit data/address during execution of some instructions.

CALL-RET instructions are used in subroutine operations. On getting a CALL in the main program, the current program counter content is pushed into the stack and loads the PC with the first memory location of the subroutine. The address of the first memory location of the subroutine is temporarily stored in W and Z registers.

Again, XCHG instruction exchanges the contents H and L with D and E respectively. W and Z registers are used for temporary storage of such data.

24. Describe the temporary data register of 8085.

Ans. The temporary data register of 8085 is an 8-bit register, which is not available to the programmer, but is used internally for execution of most of the arithmetic and logical operations.

ADD D instruction adds the contents of accumulator with the content of D. The content of D is temporarily brought into the temporary data register. Thus the two inputs to the ALU are—one from the accumulator and the other from the temporary data register. The result is stored in the accumulator.

25. Describe the general purpose registers of 8085?

Ans. The general purpose registers of 8085 are: B, C, D, E, H and L. They are all 8-bit registers but can also be used as 16-bit register pairs—BC, DE and HL. These registers are also known as scratch pad registers.

26. In what other way HL pair can be used?

Ans. HL register pair can be used as a data pointer or memory pointer.

27. Mention the utility of the general purpose registers.

Ans. General purpose registers store temporary data during program execution, which can also be stored in different accessible memory locations. But storing temporary data in memory requires bus access—hence more time is needed to store. Thus it is always advisable to store data in general purpose registers.

The more the number of general purpose registers, the more is flexibility in programming—so a microprocessor having more such registers is always advantageous.

28. Which are the sixteen bit registers of 8085.

Ans. 8085 has three (3) sixteen bit registers—Program Counter (PC), Stack Pointer (SP) and Incrementer/Decrementer address latch register.

29. Discuss the two registers program counter and stack pointer.

Ans. Program counter (PC) is a sixteen bit register which contains the address of the instruction to be executed just next. PC acts as a address pointer (also known as memory pointer) to the next instruction. As the processor executes instructions one after another, the PC is incremented—the number by which the PC increments depends on the nature of the instruction. For example, for a 1-byte instruction, PC is incremented by one, while for a 3-byte instruction, the processor increments PC by three address locations.

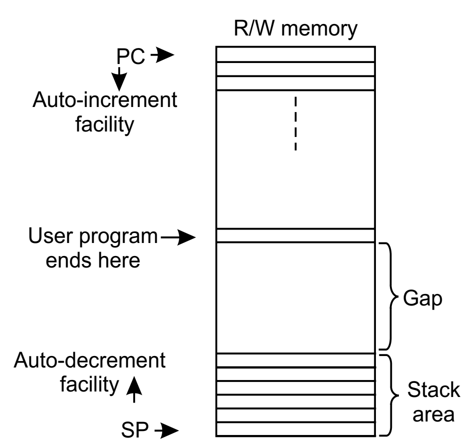


Fig. 2.5: Auto-increment and auto-decrement facility for PC and SP respectively

Stack pointer (SP) is a sixteen bit register which points to the 'stack'. The stack is an area in the R/W memory where temporary data or return addresses (in cases of subroutine CALL) are stored. Stack is a auto-decrement facility provided in the system. The stack top is initialised by the SP by using the instruction LXI SP, memory address.

In the memory map, the program should be written at one end and stack should be initialised at the other end of the map—this is done to avoid crashing of program. If sufficient

gap is not maintained between program memory location and stack, then when the stack gets filled up by PUSH or subroutine calls, the stack top may run into the memory area where program has been written. This is shown in Fig. 2.5.

30. Describe the instruction register of 8085.

Ans. Program written by the programmer resides in the R/W memory. When an instruction is being executed by the system, the opcode of the instruction is fetched from the memory and stored in the instruction register. The opcode is loaded into the instruction register during opcode fetch cycle. It is then sent to the instruction decoder.

31. Describe the (status) flag register of 8085.

Ans. It is an 8-bit register in which five bit positions contain the status of five condition flags which are Zero (Z), Sign (S), Carry (CY), Parity (P) and Auxiliary carry (AC). Each of these five flags is a 1 bit F/F. The flag register format is shown in Fig. 2.6:

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	AC	X	P	X	CY

Fig. 2.6: The flag register format

- *Sign (S) flag:* – If the MSB of the result of an operation is 1, this flag is set, otherwise it is reset.
- *Zero (Z) flag:*– If the result of an instruction is zero, this flag is set, otherwise reset.
- *Auxiliary Carry (AC) flag:*– If there is a carry out of bit 3 and into bit 4 resulting from the execution of an arithmetic operation, it is set otherwise reset.

  This flag is used for BCD operation and is not available to the programmer to change the sequence of an instruction.
- *Carry (CY) flag:*– If an instruction results in a carry (for addition operation) or borrow (for subtraction or comparison) out of bit D_7, then this flag is set, otherwise reset.
- *Parity (P) flag:*– This flag is set when the result of an operation contains an even number of 1's and is reset otherwise.

32. State the characteristics of the flag register.

Ans. The following are the characteristics of flag register:
- It is an 8-bit register.
- It contains five flags—each of one bit.
- The flag register can't be written into.

33. What is the purpose of incrementer/decrementer address latch register?

Ans. This 16-bit register increments/decrements the contents of PC or SP when instructions related to them are executed.

34. Mention the blocks on which ALU operates?

Ans. The ALU functions as a part which includes arithmetic logic group of circuits. This includes accumulator, flags F/Fs and temporary register blocks.

35. What is the function of the internal data bus?

Ans. The width of the internal data bus is 8-bit and carries instructions/data between the CPU registers. This is totally separate from the external data bus which is connected to memory chips, I/O, etc.

The internal and external data bus are connected together by a logic called a bidirectional bus (transreceiver).

36. Describe in brief the timing and control circuitry of 8085.

Ans. The T&C section is a part of CPU and generates timing and control signals for execution of instructions. This section includes Clock signals, Control signals, Status signals, DMA signals as also the Reset section. This section controls fetching and decoding operations. It also generates appropriate control signals for instruction execution as also the signals required to interface external devices.

37. Mention the following:

(*a*) **Control and Status signals**

(*b*) **Interrupt signals**

(*c*) **Serial I/O signals**

(*d*) **DMA signals**

(*e*) **Reset signals.**

Ans. The control and status signals are ALE, $\overline{RD}$, $\overline{WR}$, IO/$\overline{M}$, S_0, S_1 and READY.

The interrupt signals are TRAP, RST 7.5, RST 6.5, RST 5.5, INTR. $\overline{INTA}$ is an interrupt acknowledgement signal indicating that the processor has acknowledged an INTR interrupt.

Serial I/O signals are SID and SOD

DMA signals are HOLD and HLDA

Reset signals are $\overline{\text{RESET IN}}$ and RESET OUT.

38. What is the function of ALE and how does it function?

Ans. Pin 30 of 8085 is the ALE pin which stands for 'Address Latch Enable'. ALE signal is used to demultiplex the lower order address bus ($AD_0 – AD_7$).

Pins 12 to 19 of 8085 are $AD_0 – AD_7$ which is the multiplexed address-data bus. Multiplexing is done to reduce the number of pins of 8085.

Lower byte of address ($A_0 – A_7$) are available from $AD_0 – AD_7$ (pins 12 to 19) during T_1 of machine cycle. But the lower byte of address ($A_0 – A_7$), along with the upper byte $A_8 – A_{15}$ (pins 21 to 28) must be available during T_2 and rest of the machine cycle to access memory location or I/O ports.

Now ALE signal goes high at the beginning of T_1 of each machine cycle and goes low at the end of T_1 and remains low during the rest of the machine cycle. This high to low transition of ALE signal at the end of T_1 is used to latch the lower order address byte ($A_0 – A_7$) by the latch IC 74LS373, so that the lower byte $A_0 – A_7$ is continued to be available till the end of the machine cycle. The situation is explained in the following figure:

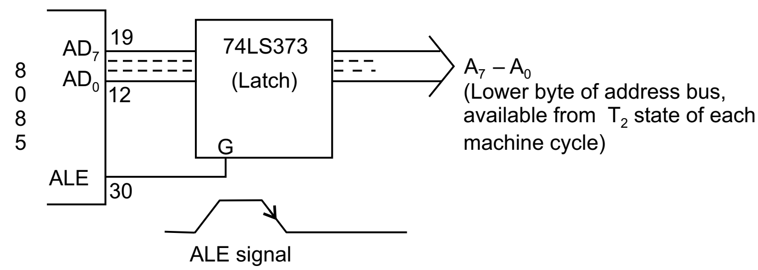


Fig. 2.7: Lower byte of address latching achieved by the H to L transition of ALE signal, which occurs at the end of T_1 of each machine cycle

39. Explain the function of the two DMA signals HOLD and HLDA.

Ans. DMA mode of data transfer is fastest and pins 39 and 38 (HOLD and HLDA) become active only in this mode.

When DMA is required, the DMA controller IC (8257) sends a 1 to pin 39 of 8085. At the end of the current instruction cycle of the microprocessor it issues a 1 to pin 38 of the controller. After this the bus control is totally taken over by the controller.

When 8085 is active and 8257 is idle, then the former is MASTER and the latter is SLAVE, while the roles of 8085 and 8257 are reversed when 8085 is idle and 8257 becomes active.

40. Discuss the three signals IO/$\overline{\text{M}}$, S_0 and S_1.

Ans. IO/$\overline{\text{M}}$ signal indicates whether I/O or memory operation is being carried out. A high on this signal indicates I/O operation while a low indicates memory operation. S_0 and S_1 indicate the type of machine cycle in progress.

41. What happens when $\overline{\text{RESET IN}}$ signal goes low?

Ans. $\overline{\text{RESET IN}}$ is an input signal which is active when its status is low. When this pin is low, the following occurs:

- The program counter is set to zero (0000_H).
- Interrupt enable and HLDA F/Fs are resetted.
- All the buses are tri-stated.
- Internal registers of 8085 are affected in a random manner.

42. Is there any minimum time required for the effective $\overline{\text{RESET IN}}$ signal?

Ans. For proper resetting to take place, the reset signal $\overline{\text{RESET IN}}$ must be held low for at least 3 clock cycles.

43. Indicate the function of RESET OUT signal.

Ans. When this signal is high, the processor is being reset. This signal is synchronised to the processor clock and is used to reset other devices which need resetting.

44. Write the advantages/disadvantages of having more number of general purpose registers in a microprocessor.

Ans. Writing of a program becomes more convenient and flexible by having more number of general purpose registers.

But there are certain disadvantages of having more GPRs. These are as follows:

The more the number of GPRs in a microprocessor, more number of bits would be required to identify individual registers. This would reduce the number of operations that can be provided by the microprocessor.

In programs involving subroutine CALL, if more GPRs are involved, then their status are to be saved in stack and on return from the subroutine, they are to be restored from the stack. This will thus put considerable overhead on the microprocessor.

If more number of GPRs are used in a microprocessor, considerable area of the chip is used up in accommodating the GPRs. Thus there may be some problem in implementing other functions on the chip.

45. Draw the lower and higher order address bus during the machine cycles.

Ans. The lower byte of address ($AD_0 - AD_7$) is available on the multiplexed address/data bus during T_1 state of each machine cycle, except during the bus idle machine cycle, shown in Fig. 2.8.

The higher byte of address ($A_8 - A_{15}$) is available during T_1 to T_3 states of each machine cycle, except during the bus idle machine cycle, shown in Fig. 2.9.

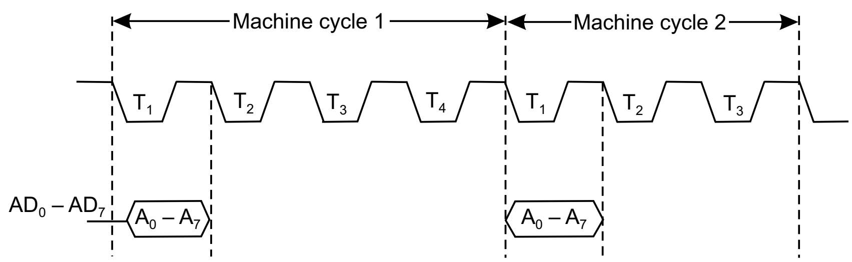


Fig. 2.8: Lower byte address on the multiplexed bus

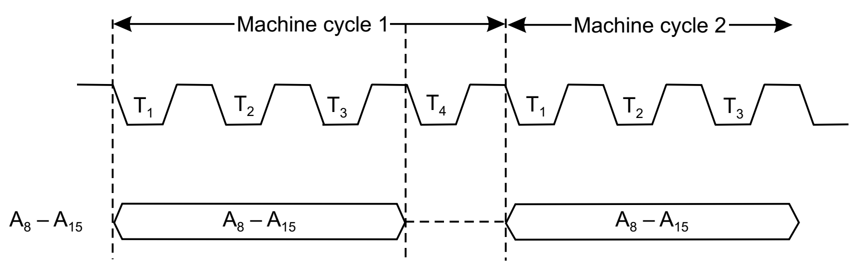


Fig. 2.9: Higher byte address on $A_8 - A_{15}$

46. Draw the appearance of data in the read and write machine cycles.

Ans. Data transfer from memory or I/O device to microprocessor or the reverse takes place during T_2 and T_3 states of the machine cycles.

In the read machine cycle, data appears at the beginning of T_3 state, whereas in the write machine cycle, it appears at the beginning of T_2, shown in Fig. 2.10.

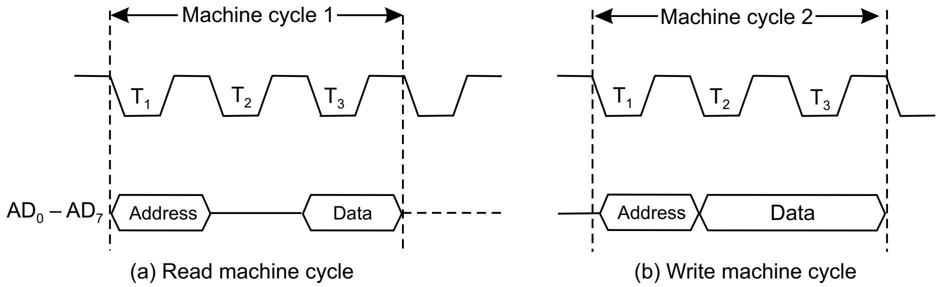


Fig. 2.10: Data bus

47. Draw the status signals during opcode fetch and memory read machine cycles.

Ans. The status signals are $IO/\overline{M}$, S_0 and S_1. Their conditions indicate the type of machine cycle that the system is currently passing through. These three status signals remain active right from the beginning till the end of each machine cycle, shown in Fig. 2.11.

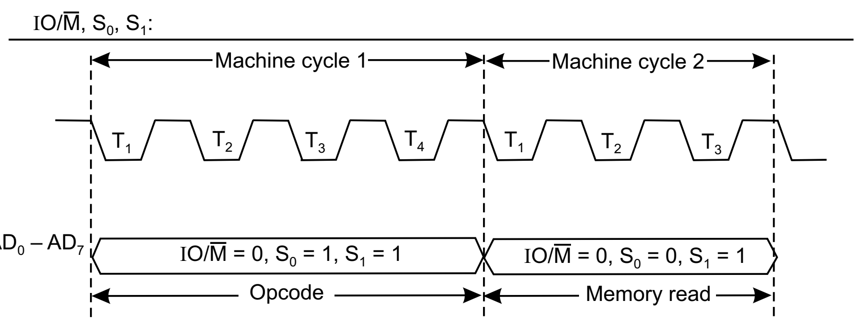


Fig. 2.11: Status signals

48. Show the $\overline{RD}$ and $\overline{WR}$ signals during the Read cycle and Write cycle.

Ans. When $\overline{RD}$ is active, microprocessor reads data from either memory or I/O device while when $\overline{WR}$ is active, it writes data into either memory or I/O device.

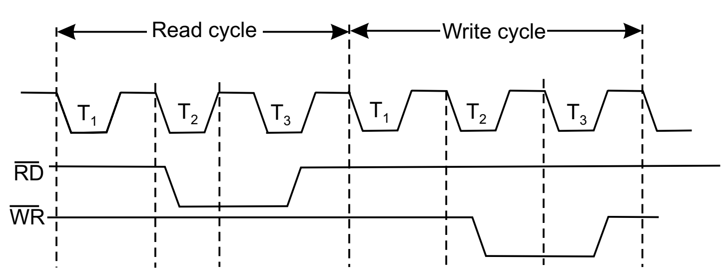


Fig. 2.12: $\overline{RD}$ and $\overline{WR}$ signals

Data transfer (reading/writing) takes place during T_2 and T_3 states of read cycle or write cycle and is shown in Fig. 2.12.

49. Indicate the different machine cycles of 8085.

Ans. 8085 has seven different machine cycles. These are:

(1) Opcode Fetch (2) Memory Read (3) Memory Write (4) I/O Read (5) I/O Write (6) Interrupt Acknowledge (7) Bus Idle.

50. Draw the Opcode Fetch machine cycle of 8085 and discuss.

Ans. The first machine cycle of every instruction is the Opcode Fetch. This indicates the kind of instruction to be executed by the system. The length of this machine cycle varies between 4T to 6T states—it depends on the type of instruction. In this, the processor places the contents of the PC on the address lines, identifies the nature of machine cycle (by IO/$\overline{M}$, S_0, S_1) and activates the ALE signal. All these occur in T_1 state.

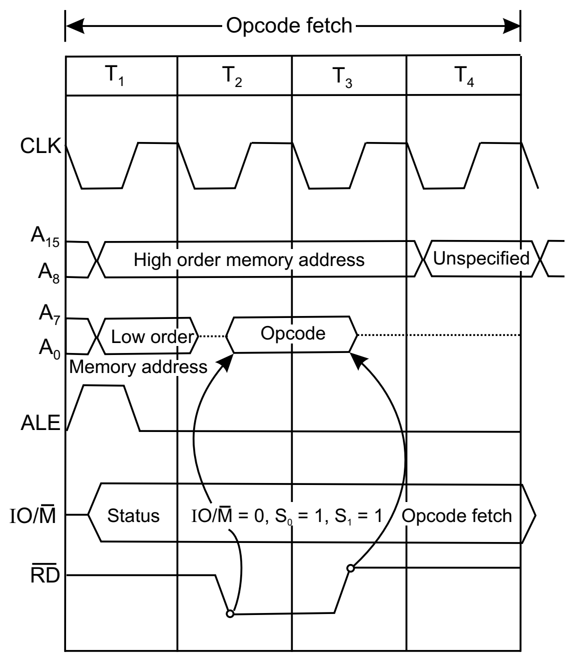


Fig. 2.13: Opcode fetch machine cycle

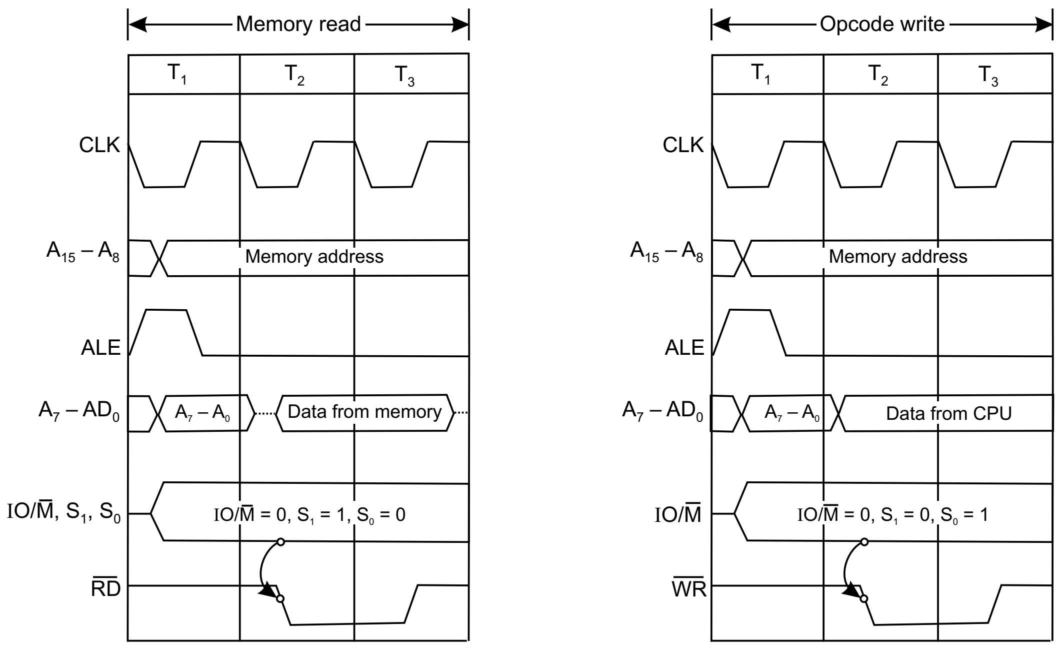
In T_2 state, $\overline{RD}$ signal is activated so that the identified memory location is read from and places the content on the data bus ($D_0 - D_7$).

In T_3, data on the data bus is put into the instruction register (IR) and also raises the $\overline{RD}$ signal thereby disabling the memory.

In T_4, the processor takes the decision, on the basis of decoding the IR, whether to enter into T_5 and T_6 or to enter T_1 of the next machine cycle.

One byte instructions that operate on eight bit data are executed in T_4. Examples are ADD B, MOV C, B, RRC, DCR C, etc.

51. Briefly describe Memory Read and Write machine cycles and show the waveforms.



(a) Memory read machine cycle (b) Memory write machine cycle

Fig. 2.14: Memory read and write machine cycle

Ans. Both the Memory Read and Memory Write machine cycles are 3T states in length. In Memory Read the contents of R/W memory (including stack also) or ROM are read while in Memory Write, it stores data into data memory (including stack memory).

As is evident from Fig. 2.14 during T_2 and T_3 states data from either memory or CPU are made available in Memory Read or Memory Write machine cycles respectively. The status signal (IO/$\overline{M}$, S_0, S_1) states are complementary in nature in Memory Read and Memory Write cycles. Reading or writing operations are performed in T_2.

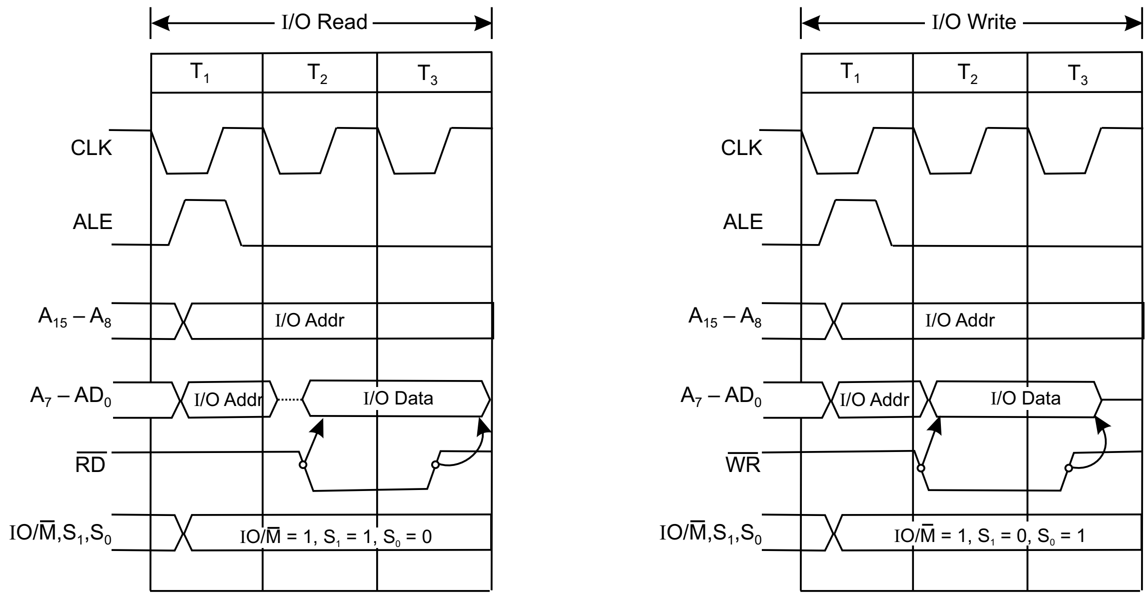
In T_3 of Memory Read, data from data bus are placed into the specified register (A, B, C, etc.) and raises $\overline{RD}$ so that memory is disabled while in T_3 of Memory Write $\overline{WR}$ signal is raised which disables the memory.

52. Draw the I/O Read and I/O Write machine cycles and discuss.

Ans. I/O Read and Write machine cycles are almost similar to Memory Read and Write machine cycles respectively. The difference here is in the IO/$\overline{M}$ signal status which remains 1 indicating that these machine cycles are related to I/O operations. These machine cycles take 3T states.

In I/O read, data are available in T_2 and T_3 states, while during the same time (T_2 and T_3) data from CPU are made available in I/O write.

The I/O read and write machine cycles are shown in Fig. 2.15.



(a) I/O read machine cycle (b) I/O write machine cycle

Fig. 2.15: I/O read and write machine cycles

53. Draw the Interrupt Acknowledge cycles for (*a*) RST instruction (*b*) CALL instruction.

Ans. The following figure shows the Interrupt Acknowledge cycle for RST instruction.

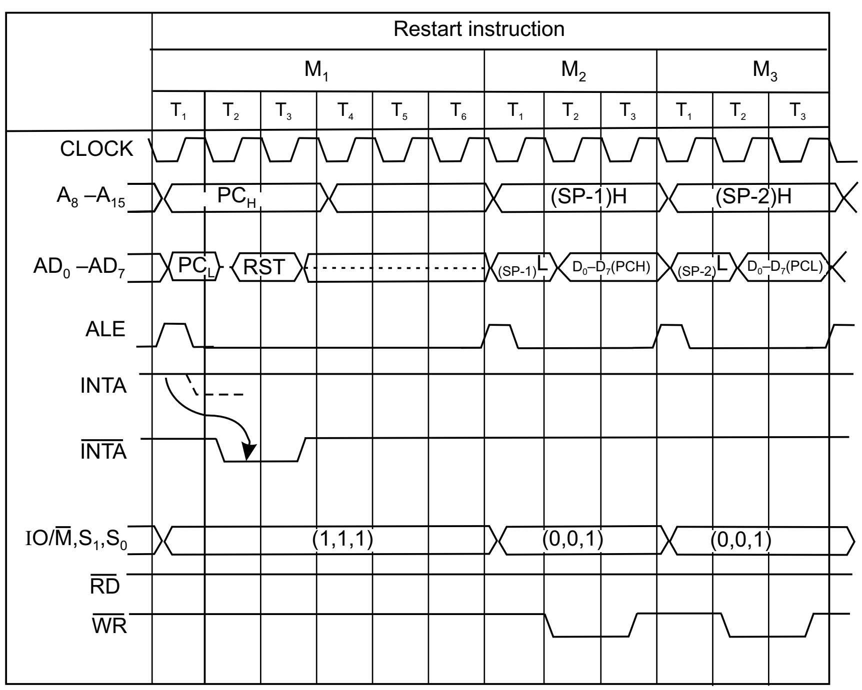


Fig. 2.16: Restart instruction

In M_1, RST is decoded. This initiates a CALL to the specific vector location. Contents of the PC are stored in stack in machine cycles M_2 and M_3.

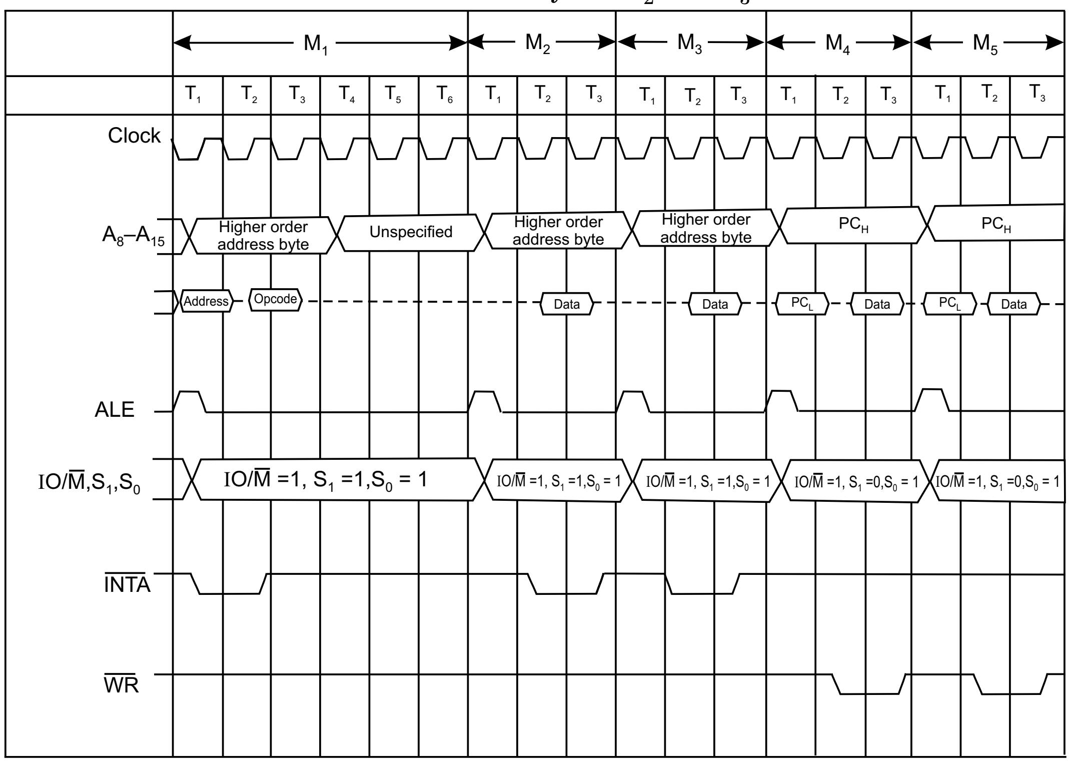


Fig. 2.17: Timing diagram of $\overline{INTA}$ machine cycle and execution of call instruction

The above figure shows an Interrupt Acknowledge cycle for CALL instruction. M_2 and M_3 machine cycles are required to call the 2 bytes of the address following the CALL. Memory write are done in machine cycles M_4 and M_5 in which contents of PC are stored in stack and then a new instruction cycle begins.

54. What is meant by Bus Idle Machine cycle?

Ans. There are a few situations in which machine cycles are neither Read or Written into. These are called Bus Idle Machine cycle.

Such situations arise when the system executes a DAD or during the internal opcode generation for the RST or TRAP interrupts.

The ALE signal changes state during T_1 of each machine cycle, but in Bus Idle Machine cycles, ALE does not change state.

55. Explain the DAD instruction and draw its timing diagram.

Ans. DAD instruction adds the contents of a specified register pair to the contents of H and L.

For execution of DAD, 10 T-states are needed. Instead of having a single machine cycle having 10 T-states, it consists of the Opcode Fetch machine cycle (4T states) and 6 extra T-states divided into two machine cycles. These two extra machine cycles are Bus Idle Machine cycles which do not involve either memory or I/O.

The timing diagram for DAD instruction is shown below:

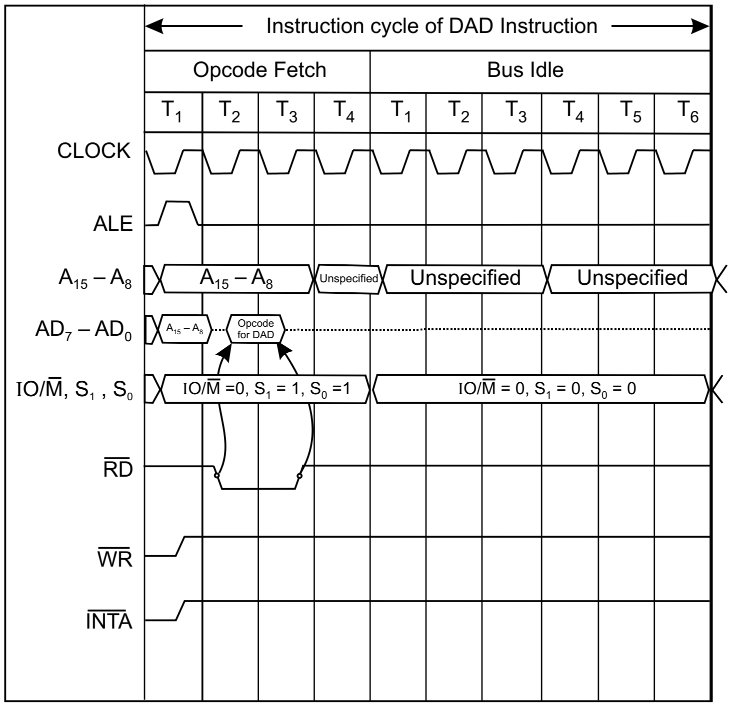


Fig. 2.18: Timing diagram for DAD instruction

56. Discuss the concept of WAIT states in microprocessors.

Ans. So many times it may happen that there is speed incompatibility between microprocessor and its memory and I/O systems. Mostly the microprocessor is having higher speed.

So in a given situation, if the microprocessor is ready to accept data from a peripheral device while there is no valid data in the device (e.g. an ADC), then the system enters into WAIT states and the READY pin (an input pin to the microprocessor, pin no. 35 for 8085) is put to a low state by the device.

Once the device becomes ready with some valid data, it withdraws the low state on the READY pin of 8085. Then 8085 accepts the data from the peripheral by software instructions.

57. Does 8085 have multiplication and division instructions?

Ans. No, 8085 does not have the above two instructions. It can neither multiply nor divide two 8-bit numbers. The same are executed by the processor following the process of repetitive addition or subtraction respectively.

58. Indicate the bus drive capability of 8085.

Ans. 8085 buses can source up to 400 μA and sink 2 mA of current. Hence 8085 buses can drive a maximum of one TTL load.

Thus the buses need bus drivers/buffers to enhance the driving capability of the buses to ensure that the voltage levels are maintained at appropriate levels and malfunctioning is avoided.

59. What are the buffers needed with the buses of 8085?

Ans. An 8-bit unidirectional buffer 74LS244 is used to buffer the higher order address bus ($A_8 - A_{15}$). It consists of eight non-inverting buffers with tri-state outputs. Each pin can sink 24 mA and source 15 mA of current.

A bidirectional buffer 74LS245 (also called octal bus transreceivers) can be used to drive the bidirectional data bus ($D_0 - D_7$) after its demultiplexing. The DIR pin of the IC controls the direction of flow of data through it.

60. Explain the instruction cycle of a microprocessor.

Ans. When a processor executes a program, the instructions (1 or 2 or 3 bytes in length) are executed sequentially by the system. The time taken by the processor to complete one instruction is called the Instruction Cycle (IC).

An IC consists of Fetch Cycle (FC) and an Execute Cycle (EC). Thus IC = FC + EC. It is shown in Fig. 2.19. Depending on the type of instruction, IC time varies.

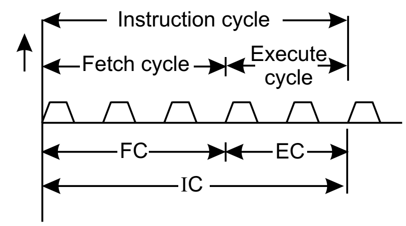


Fig. 2.19: Instruction cycle showing FC, EC and IC

61. Explain a typical fetch cycle (FC).

Ans. The time required to fetch an opcode from a memory location is called Fetch Cycle.

A typical FC may consist of 3T states. In the first T-state, the memory address, residing in the PC, is sent to the memory. The content of the addressed memory (i.e., the opcode residing in that memory location) is read in the second T-state, while in the third T-state this opcode is sent via the data bus to the instruction register (IR). For slow memories, it may take more time in which case the processor goes into 'wait cycles'. Most microprocessors have provision of wait cycles to cope with slow memories.

A typical FC may look like the following:

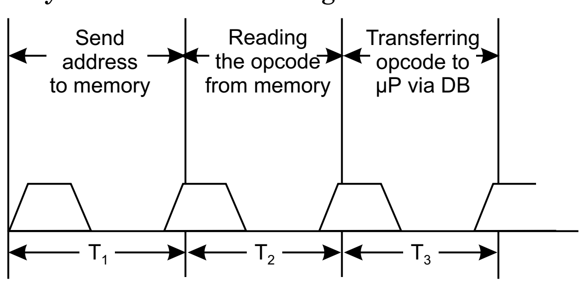


Fig. 2.20: A typical FC

62. Draw the block schematic of a typical Instruction Word flow diagram and explain the same.

Ans. There are two kinds of words—instruction word and data word. The 2 byte content of the PC is transferred to a special register–called memory address register (MAR) or simply address register (AR) at the beginning of the fetch cycle. Since the content of MAR is an address, it is thus sent to memory via the address bus. The content of the addressed memory is then read under 'Read control' generated by T&C section of the

microprocessor. This is then sent via the data bus to the memory data register (MDR) or simply data register (DR) existing in CPU. This is placed in the instruction register (IR) and is decoded by the instruction decoder and subsequently executed. The PC is then incremented if the subsequent memory location is to be accessed.

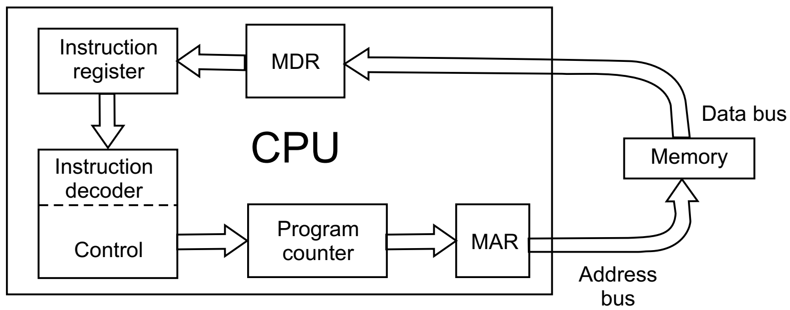


Fig. 2.21: Flow of instruction word

63. Draw the block schematic of a typical data word flow diagram and explain the same.

Ans. The data word flows via the data bus into the accumulator. The data source can be a memory device or an input device. Data from the accumulator is then manipulated in ALU under control of T & C unit. The manipulated data is then put back in the accumulator and can be sent to memory or output devices.

The block schematic of data flow is shown below.

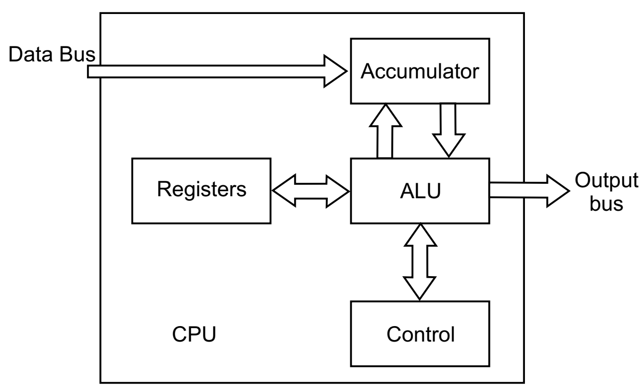


Fig. 2.22: Flow of data word

64. Why a microprocessor based system is called a sequential machine?

Ans. It can perform the jobs in a sequential manner, one after the other. That is why it is called a sequential machine.

65. Why a microprocessor based system is called a synchronous one?

Ans. All activities pertaining to the µP takes place in synchronism with the clock. Hence it is called a synchronous device.

66. Draw the diagram which will show the three buses separately, with the help of peripheral ICs.

Ans. The scheme of connections is shown below. The octal bus driver 74LS244 drives the higher order address bus $A_{15} - A_8$ while 74LS373 Latch drives the lower order address bus $A_7 - A_0$ (with the help of ALE signal). The bidirectional bus driver 74LS245 drives the data bus $D_7 - D_0$ while the 74LS138 (a 3 to 8 decoder chip) outputs at its output pins $\overline{IOW}$, $\overline{IOR}$, $\overline{MEMW}$, $\overline{MEMR}$, signals from the IO/$\overline{M}$, $\overline{RD}$ and $\overline{WR}$ signals of 8085.

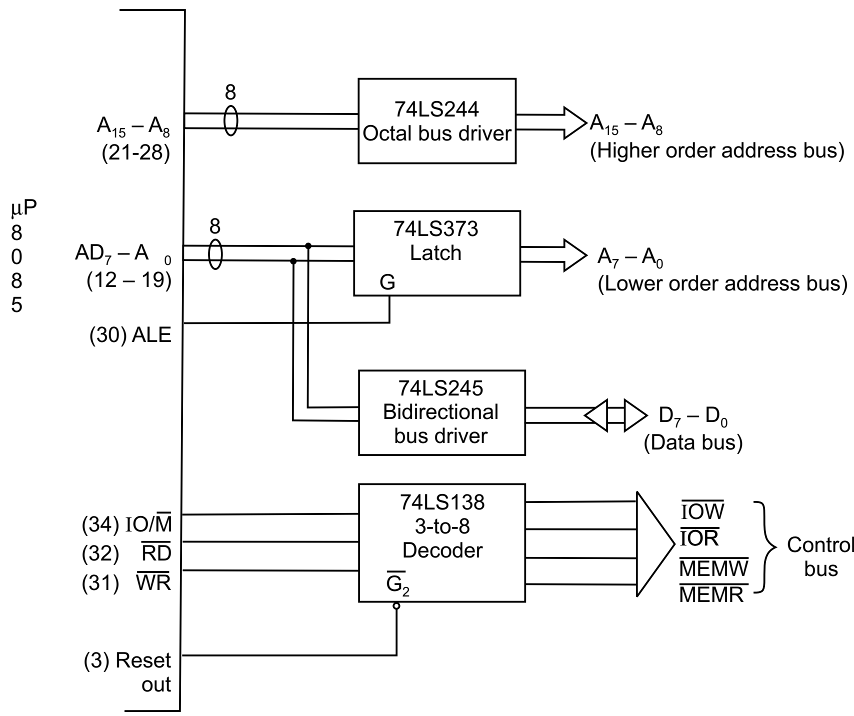


Fig. 2.23: Demultiplexing of address/data bus and separation of control signals

67. Discuss the status of $\overline{WR}$ and $\overline{RD}$ signals of 8085 at any given instant of time.

Ans. At any given instant, the status of the two signals $\overline{WR}$ and $\overline{RD}$ will be complementary to each other.

It is known that microprocessor based system is a sequential device, so that at any given point of time, it does the job of reading or writing—and definitely not the two jobs at the same time. Hence if $\overline{WR}$ signal is low, then $\overline{RD}$ signal must be high or vice versa.

68. Which registers of 8085 are programmable?

Ans. The registers B, C, D, E, H and L are programmable registers in that their contents can be changed by programming.

69. Suggest the type of operations possible on data from a look of the architecture of 8085.

Ans. The architecture of 8085 suggests that the following operations are possible.

- Can store 8-bits of data.
- Uses the temporary registers during arithmetic operations.
- Checks for the condition of resultant data (like carry, etc.) from the flag register.

70. What are the externally initiated operations supported by 8085?

Ans. 8085 supports several externally initiated operations at some of its pins. The operations possible are:

- Resetting (via Reset pin)
- Interruptions (via Trap, RST 7.5, RST 6.5, RST 5.5 and Interrupt pin)
- Ready (via Ready pin)
- Hold (via Hold pin)

71. Name the registers not accessible to the programmer.

Ans. The following registers cannot be accessed by the programmer:

- Instruction register (IR)
- Memory address register (MAR) or supply address register (AR)
- Temporary registers.

72. Name the special purpose registers of 8085.

Ans. The special purpose registers used in 8085 microprocessor are:

- Accumulator register (A)
- Program counter register (PC)
- Status (or Flag) register
- Stack pointer register (SP)

73. What should be the size of the Instruction Register if an arbitrary microprocessor has only 25 instructions?

Ans. The length of the Instruction Register would be 5-bits, since $2^5 = 32$, since a 5-bit Instruction Register can decode a maximum of 32 instructions.

74. Explain the difference between HLT and HOLD states.

Ans. HLT is a software instruction. Its execution stops the processor which enters into a HALT state and the buses of the processor are driven into tri-state.

HOLD is an hardware input to the processor. When HOLD input = 1, the processor goes into the HOLD state, but the buses don't go into tri-state. The processor gives out a high HLDA (hold acknowledge) signal which can be utilised by an external device (like a DMA controller) to take control of the processor buses. HOLD is acknowledged by the processor at the end of the current instruction execution.

75. Indicate the length of the Program Counter (PC) to access 1 KB and 1 MB memory.

Ans. 1 KB = 1024 bytes

and 1 MB = 1024 K bytes

Thus the required number of bits in PC to access 1 KB are 10 ($2^{10} = 1024$) and 20 ($2^{20} = 1024$ K) respectively.

76. What determines the number of bytes to be fetched from memory to execute an instruction?

Ans. An instruction normally consists of two fields. These are:

Opcode	Operand

Thus, while the system starts executing an instruction, it first decodes the opcode which then decides how many more bytes are to be brought from the memory—its minimum value is zero (like RAR) while the maximum value is two (like STA 4059 H).

77. A Microprocessor's control logic is 'microprogrammed'. Explain.

Ans. It implies that the architecture of the control logic is much alike the architecture of a very special purpose microprocessor.

78. Does the ALU have any storage facility?

Ans. No, it does not have any storage facility. For this reason, the need for temporary data registers arise in ALU–it has two inputs: one provided by the accumulator and the other from the temporary data register. The result of summation is stored in the accumulator.

3

Instruction Types and Timing Diagrams

1. What is an instruction?

Ans. An instruction is a command given to the microcomputer to perform a specific task or function on a given data.

2. What is meant by instruction set?

Ans. An instruction set is a collection of instructions that the microprocessor is designed to perform.

3. In how many categories the instructions of 8085 be classified?

Ans. Functionally, the instructions can be classified into five groups:

- data transfer (copy) group
- arithmetic group
- logical group
- branch group
- stack, I/O and machine control group.

4. What are the different types of data transfer operations possible?

Ans. The different types of data transfer operations possible are cited below:

- Between two registers.
- Between a register and a memory location.
- A data byte can be transferred between a register and a memory location.
- Between an I/O device and the accumulator.
- Between a register pair and the stack.

The term 'data transfer' is a misnomer—actually data is not transferred, but copied from source to destination.

5. Mention the different types of operations possible with arithmetic, logical, branch and machine control operations.

Ans. The arithmetic operations possible are addition, subtraction, increment and decrement.

The logical operations include AND, OR, EXOR, compare, complement, while branch operations are Jump, Call, Return and Restart instructions.

The machine control operations are Halt, Interrupt and NOP (no operation).

6. What are the different instruction word sizes in 8085?

Ans. The instruction word sizes are of the following types:

- 1-byte instruction
- 2-byte instruction
- 3-byte instruction.

7. What an instruction essentially consists of?

Ans. An instruction comprises of an operation code (called 'opcode') and the address of the data (called 'operand'), on which the opcode operates. This is the structure on which an instruction is based. The opcode specifies the nature of the task to be performed by an instruction. Symbolically, an instruction looks like

Operation code	Address of data
opcode	operand

8. Give one example each of 1-byte, 2-byte and 3-byte instructions.

Ans. The examples are given below:

- 1-byte instruction : ADD B
- 2-byte instruction : MVIC, 07
- 3-byte instruction : LDA 4400

In 1-byte instruction, the opcode and the operand are in the same byte i.e.,

Opcode/Operand

A 2-byte instruction looks like this:

Opcode	1st byte
Operand or data/address	2nd byte

While a 3-byte instruction looks like the following:

Opcode	1st byte
Low order byte of address	2nd byte
High order byte of address	3rd byte

9. What is meant by 'addressing mode'? Mention the different addressing modes.

Ans. Each instruction indicates an operation to be performed on certain data. There are various methods to specify the data for the instructions, known as 'addressing modes'.

For 8085 microprocessor, there are five addressing modes. These are:

- Direct addressing
- Register addressing
- Register indirect addressing
- Immediate addressing
- Implicit addressing.

10. Give one example each of the five types of addressing modes.

Ans. The examples for each type of addressing mode are given below:

(*a*) *Direct Addressing:* In this mode, the operand is specified within the instruction itself.

Examples of this type are:

LDA 4000_H, STA 5513_H, etc.

IN/OUT instructions (like IN PORT C, OUT PORT B, etc.) also falls under this category.

(*b*) *Register Addressing:* In this mode of addressing, the operand are in the general purpose registers.

Examples are: MOV A, B ; ADD D, etc.

(*c*) *Register Indirect Addressing:* MOV A, M; ADD M are examples of this mode of addressing. These instructions utilise 1-byte. In this mode, instead of specifying a register, a register pair is specified to accommodate the 16-bit address of the operand.

(*d*) *Immediate Addressing:* MVI A, 07; ADI 0F are examples of Immediate Addressing mode.
The operand is specified in the instruction in this mode. Here, the operand address is not specified.

(*e*) *Implicit Addressing:* In this mode of addressing, the operand is fully absent. Examples are RAR, RAL, CMA, etc.

11. Let at the program memory location 4080, the instruction MOV B, A (opcode 47_H) is stored while the accumulator content is FF_H. Illustrate the execution of this instruction by timing diagram.

Ans. Since the program counter sequences the execution of instructions, it is assumed that the PC holds the address 4080_H. While the system executes the instruction, the following takes place one after another.

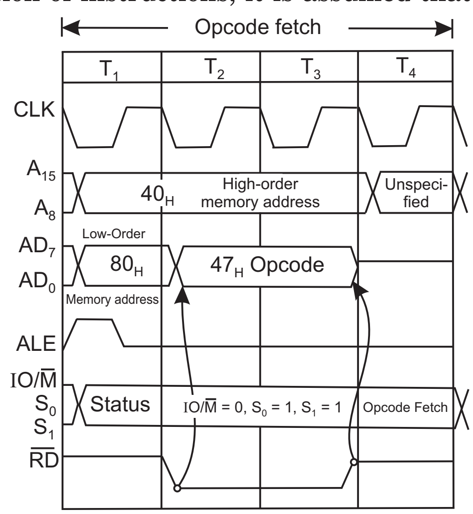


Fig. 3.1: 8085 timing for execution of the instruction (MOV B, A)

1. The CPU places the address 4080_H (residing in PC) on the address bus—40_H on the high order bus $A_{15} – A_8$ and 80_H on the low order bus $AD_7 – AD_0$.
2. The CPU raises the ALE signal to go high—the H to L transition of ALE at the end of the first T state demultiplexes the low order bus.
3. The CPU identifies the nature of the machine cycle by means of the three status signals IO/$\overline{M}$, S_0 and S_1.

   IO/ = 0, S_1 = 1, S_0 = 1
4. In T_2, memory is enabled by the signal. The content of PC i.e., 47_H is placed on the data bus. PC is incremented to 4081_H.
5. In T_3, CPU reads 47_H and places it in the instruction register.
6. In T_4, CPU decodes the instruction, places FF_H (accumulator content) in the temporary register and then transfers it to register B. Figure 3.1 shows the execution of the above. It consists of 4T states.

4

8085 Interrupts

1. Mention the interrupt pins of 8085.

Ans. There are five (5) interrupt pins of 8085—from pin 6 to pin 10. They represent TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR interrupts respectively. These five interrupts are 'hardware' interrupts.

2. Explain maskable and non-maskable interrupts.

Ans. An interrupt which can be disabled by software means, is called a maskable interrupt. Thus an interrupt which cannot be masked is an unmaskable interrupt.

3. Which is the non-maskable interrupt for 8085?

Ans. TRAP interrupt is the non-maskable interrupt for 8085. It means that if an interrupt comes via TRAP, 8085 will have to recognise the interrupt.

4. Do the interrupts of 8085 have priority?

Ans. Yes, the interrupts of 8085 have their priorities fixed—TRAP interrupt has the highest priority, followed by RST 7.5, RST 6.5, RST 5.5 and lastly INTR.

5. What is meant by priority of interrupts?

Ans. It means that if 8085 is interrupted by more than one interrupt at the same time, the one which is having highest priority will be serviced first, followed by the one(s) which is (are) having just next priority and so on.

For example, if 8085 is interrupted by RST 7.5, INTR and RST 5.5 at the same time, then the sequence in which the interrupts are going to be serviced are as follows: RST 7.5, RST 5.5 and INTR respectively.

6. Mention the types of interrupts that 8085 supports.

Ans. 8085 supports two types of interrupts—hardware and software interrupts.

7. What are the software interrupts of 8085? Mention the instructions, their hex codes and the corresponding vector addresses.

Ans. 8085 has eight (8) software interrupts from RST 0 to RST 7. The instructions, hex codes and the vector locations are tabulated in Table 4.1:

Table 4.1: Vector addresses for software interrupts

Instruction	Corresponding HEX code	Vector addresses
RST 0	C7	0000H
RST 1	CF	0008H
RST 2	D7	0010H
RST 3	DF	0018H
RST 4	E7	0020H
RST 5	EF	0028H
RST 6	F7	0030H
RST 7	FF	0038H

8. How the vector address for a software interrupt is determined?

Ans. The vector address for a software interrupt is calculated as follows:

Vector address = interrupt number × 8

For example, the vector address for RST 5 is calculated as

$5 \times 8 = 40)_{10} = 28)_H$

∴ Vector address for RST 5 is 0028_H.

9. In what way INTR is different from the other four hardware interrupts?

Ans. There are two differences, which are discussed below:

1. While INTR is not a vectored interrupt, the other four, viz., TRAP, RST 7.5, RST 6.5 and RST 5.5 are all vectored interrupts. Thus whenever an interrupt comes via any one of these four interrupts, the internal control circuit of 8085 produces a CALL to a predetermined vector location. At these vector locations the subroutines are written.
   On the other hand, INTR receives the address of the subroutine from the external device itself.
2. Whenever an interrupt occurs via TRAP, RST 7.5, RST 6.5 or RST 5.5, the corresponding returns address (existing in program counter) is auto-saved in STACK, but this is not so in case of INTR interrupt.

10. Indicate the nature of signals that will trigger TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.

Ans. TRAP interrupt is both positive edge and level triggered, RST 7.5 is positive edge triggered while RST 6.5, RST 5.5 and INTR are all level triggered.

11. Why the TRAP input is edge and level sensitive?

Ans. TRAP input is edge and level sensitive to avoid false triggering caused by noise and transients.

12. Draw the TRAP interrupt circuit diagram and explain the same.

Ans.

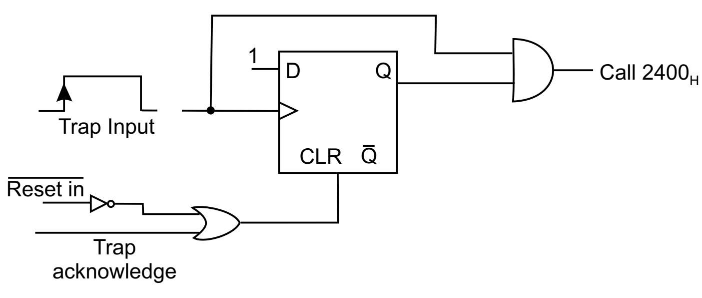


Fig. 4.1: The TRAP interrupt circuit

The positive edge of the TRAP signal sets the D F/F, so that Q becomes 1. It thus enables the AND gate, but the AND gate will output a 1 for a sustained high level at the TRAP input. In that case the subroutine written at vector memory location 2400_H corresponding to TRAP interrupt starts executing. 2400_H is the starting address of an interrupt service routine for TRAP.

There are two ways to clear the TRAP interrupt:

1. When the microprocessor is resetted, then via the inverter, a high comes out of the OR gate, thereby clearing the D F/F, making Q = 0.
2. When a TRAP is acknowledged by the system, an internal TRAP ACKNOWLEDGE is generated thereby clearing the D F/F.

13. Discuss the INTR interrupt of 8085.

Ans. The following are the characteristics of INTR interrupt of 8085:

- It is a maskable interrupt
- It has lowest priority
- It is a non-vectored interrupt.

Sequentially, the following occurs when INTR signal goes high:

1. 8085 checks the status of INTR signal during execution of each instruction.
2. If INTR signal remains high till the completion of an instruction, then 8085 sends out an active low interrupt acknowledge ($\overline{INTA}$) signal.
3. When $\overline{INTA}$ signal goes low, external logic places an instruction OPCODE on the data bus.
4. On receiving the instruction, 8085 saves the address of next instruction (which would have otherwise been executed) in the STACK and starts executing the ISS (interrupt service subroutine).

14. Draw the diagram that outputs RST 3 instruction opcode on acknowledging the interrupt.

Ans. The diagram is shown below:

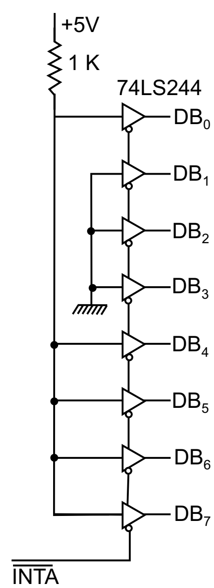


Fig. 4.2: Hardware circuit diagram to implement RST 3 (Opcode DF H)

When an INTR is acknowledged by the microprocessor, it outputs a low $\overline{\text{INTA}}$. Thus an RST 3 is gated onto the system bus. The processor then first saves the PC in the STACK and branches to the location 0018_H. At this address, the ISS begins and ends with a RET instruction. On encountering RET instruction in the last line of the ISS, the return address saved in the stack is restored in the PC so that normal processing in the main program (at the address which was left off when the program branched to ISS) begins.

15. What is to be done if a particular part of a program is not to be interrupted by RST 7.5, RST 6.5, RST 5.5 and INTR?

Ans. Two software instructions—EI and DI are used at the beginning and end of the particular portion of the program respectively. The scheme is shown schematically as follows:

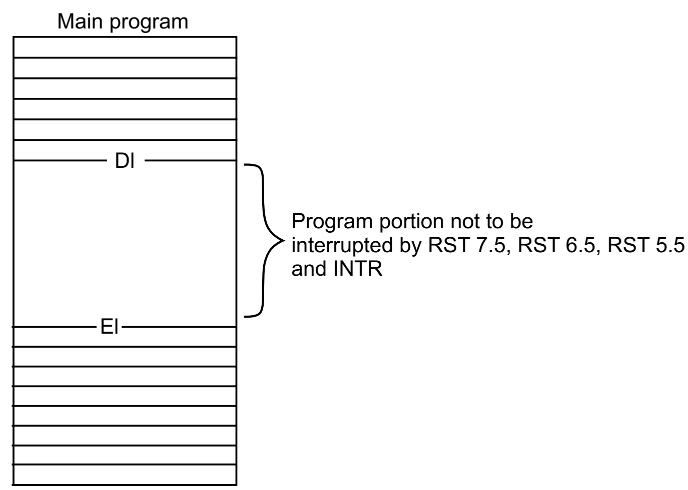


Fig. 4.3: Employing EI and DI in a program

16. Explain the software instructions EI and DI.

Ans. The EI instruction sets the interrupt enable flip-flop, thereby enabling RST 7.5, RST 6.5, RST 5.5 and INTR interrupts.

The DI instruction resets the interrupt enable flip-flop, thereby disabling RST 7.5, RST 6.5, RST 5.5 and INTR interrupts.

17. When returning back to the main program from Interrupt Service Subroutine (ISS), the software instruction EI is inserted at the end of the ISS. Why?

Ans. When an interrupt (either via RST 7.5, RST 6.5, RST 5.5, INTR) is acknowledged by the microprocessor, 'any interrupt acknowledge' signal resets the interrupt enable F/F. It thus disables RST 7.5, RST 6.5, RST 5.5 and INTR interrupts. Thus any future interrupt coming via RST 7.5, RST 6.5, RST 5.5 or INTR will not be acknowledged unless the software instruction EI is inserted which thereby sets the interrupt enable F/F.

18. Mention the ways in which the three interrupts RST 7.5, RST 6.5 and RST 5.5 are disabled?

Ans. The three interrupts can be disabled in the following manner:

- Software instruction DI
- $\overline{\text{RESET IN}}$ signal
- Any interrupt acknowledge signal.

19. Draw the SIM instruction format and discuss.

Ans. The SIM instruction format is shown below:

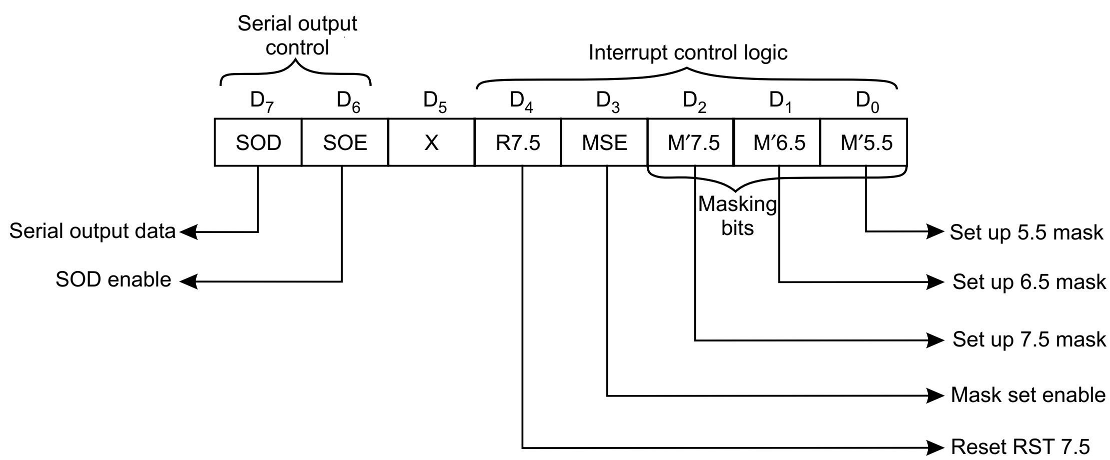


Fig. 4.4: The SIM instruction format

D_7 and D_6 bits are utilised for serial outputting of data from accumulator. D_5 bit is a don't care bit, while bits D_4–D_0 are used for interrupt control.

D_4 bit can clear the D F/F associated with RST 7.5.

D_3 bit is mask set enable (MSE) bit, while bits D_2–D_0 are the masking bits corresponding to RST 7.5, RST 6.5 and RST 5.5 respectively.

None of the flags are affected by SIM instruction.

By employing SIM instruction, the three interrupts RST 7.5, RST 6.5 and RST 5.5 can be masked or unmasked. For masking any one of these three interrupts, MSE (i.e., bit D_3) bit must be 1.

For example let RST 7.5 is to be masked (disabled), while RST 6.5 and RST 5.5 are to be unmasked (enabled), then the content of the bits of the SIM instruction will be like

$$0000\ 1100 = 0C_H$$

For this to be effective the following two instructions are written,

MVI A, $0C_H$

SIM

Execution of SIM instruction allows copying of the contents of the accumulator into the interrupt masks.

20. Show the RIM instruction format and discuss the same.

Ans. RIM stands for 'Read interrupt mask' and its format is as follows:

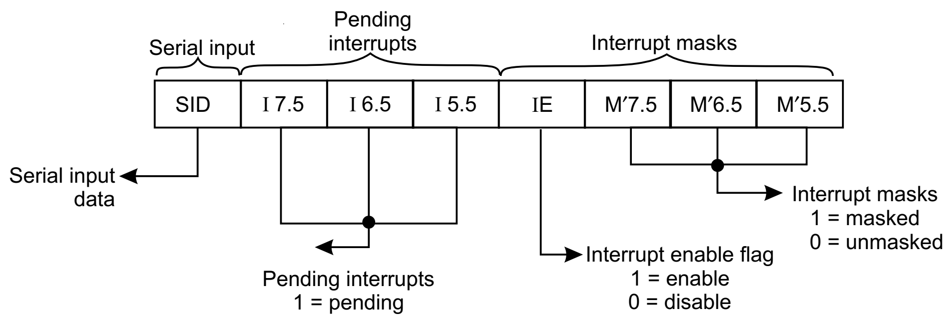


Fig. 4.5: The RIM instruction format

When RIM instruction is executed in software, the status of SID, pending interrupts and interrupt masks are loaded into the accumulator. Thus their status can be monitored. It may so happen that when one interrupt is being serviced, other interrupt(s) may occur. The status of these pending interrupts can be monitored by the RIM instruction. None of the flags are affected by RIM instruction.

21. Write a program which will call the interrupt service subroutine (at 3C00H) corresponding to RST 7.5 if it is pending. Let the ACC content is 20 H on executing the RIM instruction.

Ans. The program for the above will be as hereunder:

		SID	I7.5	I6.5	I5.5	IE	M′7.5	M′6.5	M′5.5
RIM	⇒ ACC	0	0	1	0	0	0	0	0
ANI 40 H	⇒ AND immediate with 40 H	0	1	0	0	0	0	0	0

Isolate 17.5 bit

CNZ 3C00H ⇒ Call interrupt service subroutine corresponding to RST 7.5, if RST 7.5 is pending.

22. Write a program which will call the subroutine (say named 'SR') if RST 6.5 is masked. Let content of ACC is 20 H on executing the RIM instruction.

Ans. RST 6.5 is masked if bit M′6.5 (D_1 bit of RIM) is a 1 and also D_3 bit (i.e., IE) is 1

The program for the above will be as hereunder:

		SID	I7.5	I6.5	I5.5	IE	M′7.5	M′6.5	M′5.5
RIM	⇒ ACC	0	0	1	0	0	0	0	0
ANI 0A H	⇒ AND immediate with 0A H	0	0	0	0	1	0	1	0

Isolate M′6.5 bit

JNZ SR ⇒ Jump to subroutine "SR" if RST 6.5 is masked.

23. For what purpose TRAP interrupt is normally used?

Ans. TRAP interrupt is a non-maskable one i.e., if an interrupt comes via the TRAP input, the system will have to acknowledge that. That is why it is used for vital purposes which require immediate attention like power failure.

If the microprocessor based system loses power, the filter capacitors hold the supply voltage for several mili seconds.

During this time, data in the RAM can be written in a disk or E^2PROM for future usage.

24. Draw the interrupt circuit diagram for 8085 and explain.

Ans. Figure 4.6 is the interrupt circuit diagram of 8085. It shows the five hardware interrupts TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR along with the software interrupts RST n: n = 0 to 7.

Trap is both edge and level sensitive interrupt. A short pulse should be applied at the trap input, but the pulse width must be greater than a normal noise pulse width and also long enough for the µP to complete its current instruction. The trap input must come down to low level for it to be recognised for the second time by the system. It is having highest priority.

Next highest priority interrupt is RST 7.5 which responds to the positive edge (low to high transition) of a pulse. Like trap, it also has a D F/F whose output becomes 1 on accepting the RST 7.5 input, but final call to vector location 3C00H is reached only if RST 7.5 remains unmasked and the program has an EI instruction inserted already. These are evident from the circuit. If R 7.5 (bit D_4 of SIM instruction) is 1, then RST 7.5 instruction will be overlooked i.e., it can override any RST 7.5 interrupt.

Like RST 7.5, final call locations 3400_H and $2C00_H$ corresponding to interrupts at RST 6.5 and RST 5.5 are reached only if the two interrupts remain unmasked and the software instruction EI is inserted.

The three interrupts RST 7.5, RST 6.5 and RST 5.5 are disabled once the system accepts an interrupt input via any one of these pins—this is because of the generation of 'any interrupt acknowledge' signal which disables them.

Any of the software RST instructions (RST n : n = 0 to 7) can be utilised by using INTR instruction and hardware logic. RST instructions are utilised in breakpoint service routine to check register(s) or memory contents at any point in the program.

25. The process of interrupt is asynchronous in nature. Why?

Ans. Interrupts may come and be acknowledged (provided masking of any interrupt is not done) by the microprocessor without any reference to the system clock. That is why interrupts are asynchronous in nature.

26. In how many categories can 'interrupt requests' be classified?

Ans. The 'interrupt requests' can be classified into two categories—maskable interrupt and non-maskable interrupt.

A maskable interrupt can either be ignored or delayed as per the needs of the system while a non-maskable interrupt has to be acknowledged.

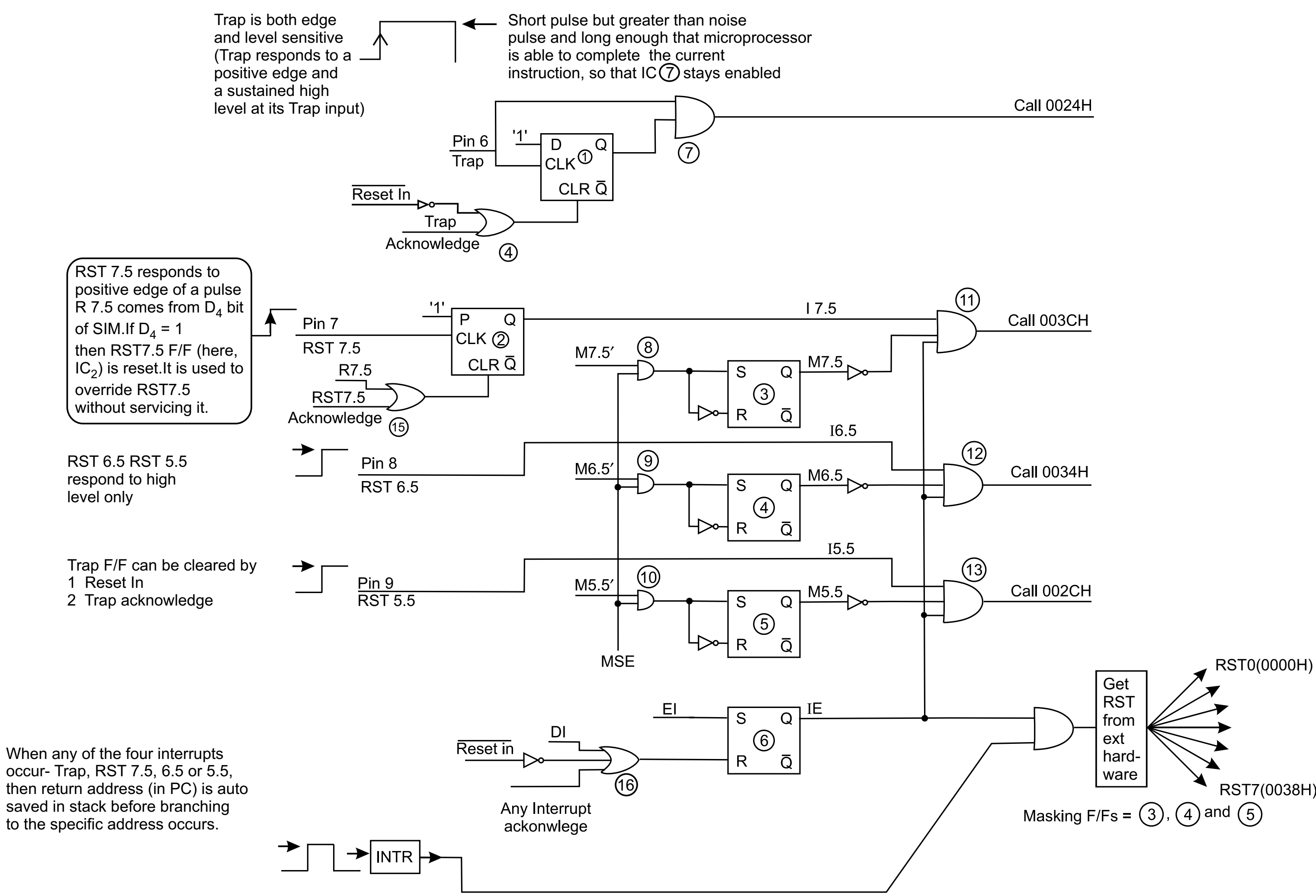


Fig. 4.6: Interrupt circuit description

27. When the interrupt pins of 8085 are checked by the system?

Ans. Microprocessor checks (samples) interrupt pins one cycle before the completion of an instruction cycle, on the falling edge of the clock pulse. An interrupt occurring at least 160 ns (150 ns for 8085A-2, since it is faster in operation) before the sampling time is treated as a valid interrupt.

28. Is there a minimum pulse width required for the INTR signal?

Ans. Microprocessor issues a low $\overline{\text{INTA}}$ signal as an acknowledgement on receiving an INTR interrupt input signal. A CALL instruction is then issued so that the program branches to Interrupt Service Subroutine (ISS). Now the CALL requires 18 T-states to complete. Hence the INTR pulse must remain high for at least 17.5 T-states. If 8085 is operated at 3 MHz clock frequency, then the INTR pulse must remain high for at least 5.8 μS.

29. Can the microprocessor be interrupted before completion of existing Interrupt Service Subroutine (ISS)?

Ans. Yes, the microprocessor can be interrupted before the completion of the existing ISS.

Let after acknowledging the INTR, the microprocessor is in the ISS executing instructions one by one. Now, for a given situation, if the interrupt system is enabled (by inserting an EI instruction) just after entering the ISS, the system can be interrupted again while it is in the first ISS.

If an interrupt service subroutine be interrupted again then this is called 'nested interrupt'.

30. Bring out one basic difference between SIM and DI instructions.

Ans. While by using SIM instruction any combinations or all of RST 7.5, RST 6.5 and RST 5.5 can be disabled, on the other hand DI disables RST 7.5, RST 6.5, RST 5.5 and in addition INTR interrupt also.

31. What is RIM instruction and what does it do?

Ans. The instruction RIM stands for Read Interrupt Mask. By executing this instruction in software, it is possible to know the status of interrupt mask, pending interrupt(s), and serial input.

32. In an interrupt driven system, EI instruction should be incorporated at the beginning of the program. Why?

Ans. A program, written by a programmer in the RAM location, is started first by system reset and loading the PC with the starting address of the program.

Now, with a system reset, all maskable interrupts are disabled. Hence, an EI instruction must be put in at the beginning of the program so that the maskable interrupts, which should remain unmasked in a program, remain so.

33. How the system can handle multiple interrupts?

Ans. Multiple interrupts can be handled if a separate interrupt is allocated to each peripheral.

The programmable interrupt controller IC 8259 can also be used to handle multiple interrupts when they are interfaced through INTR.

34. When an interrupt is acknowledged, maskable interrupts are automatically disabled. Why?

Ans. This is done so that the interrupt service subroutine (ISS) to which the program has entered on receiving the interrupt, has a chance to complete its own task.

35. What is meant by 'nested interrupts'? What care must be taken while handling nested interrupts?

Ans. Interrupts occurring within interrupts are called 'nested interrupts'.

While handling nested interrupts, care must be taken to see that the stack does not grow to such an extent as to foul the main program—in that case the system program fails.

36. 'A RIM instruction should be performed immediately after TRAP occurs'—Why?

Ans. This is so as to enable the pre-TRAP interrupt status to be restored with the implementation of a SIM instruction.

37. What does the D_4 bit of SIM do?

Ans. Bit D_4 of SIM is R 7.5 which is connected to RST 7.5 F/F via a OR gate. If D_4 of SIM is made a 1, then it resets RST 7.5 F/F. This thus can be used to override RST 7.5 without servicing it.

38. Comment on the TRAP input of 8085.

Ans. Trap input is both edge and level sensitive. It is a narrow pulse, but the pulse width should be more than normal noise pulse width. This is done so that noise cannot affect the TRAP input with a false triggering. Again the pulse width should be such that the TRAP input which is directly connected to the gate stays high till the completion of current instruction by the μP. In that case, only the program gets diverted to vector call location 2400 H.

TRAP cannot respond for a second time until the first TRAP goes through a high to low transition.

TRAP interrupt, once acknowledged, goes to 2400 H vector location without any external hardware or EI instruction, as is the case for other interrupt signals to be acknowledged.

39. Discuss about the triggering levels of RST 7.5, RST 6.5 and RST 5.5.

Ans. RST 7.5 is positive edge sensitive and responds to a short trigger pulse. The interrupt that comes via RST 7.5 is stored in a D F/F, internal to μP. The final vector call location 3C00 H is invoked only if RST 7.5 remains unmasked via SIM and software instruction EI is inserted in the program.

RST 6.5 and 5.5 respond to high level (i.e., level sensitive) at their input pins—thus these two pins must remain high until microprocessor completes the execution of the current instruction.

40. Discuss the utility of RST software instruction.

Ans. For an RST instruction (RST n, n : 0 to 7) to become effective, external hardware is necessary, along with INTR interrupt instruction.

When debugging is required in a program to know the register(s) or memory contents, breakpoints are inserted via RST instruction.

A breakpoint is an RST instruction inserted in a program. When an RST instruction is recognised, the program control is transferred to the corresponding RST vector location. From this vector location, it is again transferred to the breakpoint service routine so that programmer can check the contents of any register or memory content on pressing specified key(s). After testing, the routine returns to the breakpoint in the main program.

Thus RST instructions can be inserted within a program to examine the register/ memory content as per the requirement.

41. Under what condition, an RST instruction is going to be recognised?

Ans. Any RST instruction is recognised only if the EI instruction is incorporated via software.

42. Can the 'TRAP' interrupt be disabled by a combination of hardware and software?

Ans. Yes, it can be disabled by SIM instruction and hardware, as shown in Fig. 4.7.

The following two instructions are executed.

MVIA, 40 H
SIM

It ensures that a '0' logic comes out via SOD pin (pin 4) of 8085. This is then ANDed with TRAP input.

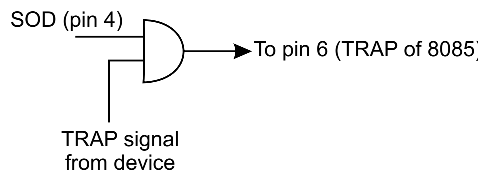


Fig. 4.7

Thus pin 6 (TRAP) always remains at '0' logic and hence TRAP input is disabled or 'MASKED'.

43. Level wise, how the interrupts can be classified? Distinguish them.

Ans. Level wise, interrupts can be classified as

• single level interrupts
• multi level interrupts

Their distinguishing features are shown below:

Single level interrupt	Multi level interrupt
1. Interrupts are fed via a single pin of microprocessor (like INTR of 8085)	1. Interrupts are fed via different pins of microprocessor (like RST 7.5, RST 6.5 etc), each interrupt requiring a separate pin of microprocessor.
2. CPU polls the I/O devices to identify the interrupting device.	2. Since each interrupt pin corresponds to a single I/O device, polling is not necessary.
3. Slower because of sl. no. 2.	3. Faster because of sl. no. 2.

5

Programming Techniques

1. Explain DAA instruction.

Ans. It stands for 'decimal adjust accumulator'.

Execution of DAA instruction converts the content of the accumulator into two BCD values. The system utilises the AC flag (not accessible by the programmer, but is used internally for DAA operation) for this conversion by following the procedures stated below:

(*a*) If the lower order 4-bits ($D_3 - D_0$) of the accumulator is greater than 9_{10} or if the AC flag is set, then this instruction (i.e., DAA) adds 06_{10} to the low-order 4-bits.

(*b*) If the higher order 4-bits ($D_7 - D_4$) of the accumulator is greater than 9_{10} or if the CY flag is set, then this instruction (i.e., DAA) adds 60_{10} to the high-order 4-bits.

Examples follow to explain the above:

(*i*) Let ACC contains 34_{BCD}. Add 19_{BCD} to this

34_{BCD}	=	0	0	1	1	0	1	0	0	
19_{BCD}	=	0	0	0	1	1	0	0	1	
53_{BCD}	=	0	1	0	0	1	1	0	1	= 4D H

Since the lower 4-bits represent D (> 9_{10}), hence execution of DAA adds 06_{10} to the above

4D	=	0	1	0	0	1	1	0	1	
06	=	0	0	0	0	0	1	1	0	
		0	1	0	1	0	0	1	1	= 53_{BCD}

(*ii*) Let ACC = 99_{BCD}. Add 79_{BCD} to this

99_{BCD}	=	1	0	0	1	1	0	0	1	
79_{BCD}	=	0	1	1	1	1	0	0	1	
178_{BCD}	=	0	0	0	1	0	0	1	0	= 12 H
		1				1				
		CY				AC				

Here both higher order (0001) and lower order (0010) 4-bits are less than 9_{10}, but both AC and CY flags are set. Thus, DAA instruction execution will add 66_{10} to the result.

12 =	0	0	0	1	0	0	1	0	
66 =	0	1	1	0	0	1	1	0	
	0	1	1	1	1	0	0	0	$= 78_{BCD}$

and since CY is already set, thus the answer is 178_{BCD}.

2. What happens when HLT is executed in software?

Ans. All the buses go into the tri-state on execution of HLT instruction.

3. Write down the instructions that load H-L register pair by the contents of memory location 3500 H. Then move the contents to register C.

Ans. The corresponding instructions are

	LXI H, 3500 H	⇒	execution of this instruction loads H-L register pair with the contents of memory location 3500 H.
and	MOV C, M	⇒	execution of this instruction moves the contents of memory location 3500 H to register C.

4. Explain the two instructions (*a*) LDAX and (*b*) STAX

Ans. (*a*) The instruction LDAX indicates that the contents of the designated register pair point to a memory location and copies the contents of the memory location into the accumulator.

As an example, let D = 40 H, E = 50 H and memory location 4050 H = AB H. Then LDAX D transfers the contents of memory location 4050 H to the accumulator. Thus, after the execution of instruction, ACC = AB H

Register contents before instruction

A	XX	XX	F
D	40	50	E

Memory location

4050	AB

Register contents after instruction

A	AB	XX	F
D	40	50	E

(*b*) STAX stands for 'store accumulator indirect'. The contents of the accumulator are copied into the memory location specified by the contents of the register pair.

As an example, let D = 40 H and E = 50 H and ACC = AB H. Then STAX D stores the accumulator contents in the memory location 4050 H.

Register contents before instruction

A	AB	XX	F
D	40	50	E

Accumulator Content

ACC = AB H

Register and memory contents after instruction

A	AB	XX	F
D	40	50	E

4050 = AB H

5. Explain the instructions (*a*) SHLD (*b*) LHLD.

Ans. (*a*) It stands for 'Store H and L registers direct'. The instruction SHLD is followed by a 2-byte address. The content of L is stored in this address and the content of H is stored in the memory location that follows.

As an example, let H = 40 H and L = 50 H. Then execution of SHLD 6555 H stores 50 H in memory location 6555 H and 40 H is stored in 6556 H memory address.

Register contents before instruction

H	40	50	L

Memory and register contents after instruction

6555	50	H	40	50	L
6556	40				

(*b*) It stands for 'Load H and L registers direct'.
It copies the contents of the memory location (that follows an LHLD instruction) into L register and copies the contents of the next memory location in H register. The contents of the two accessed memory locations remain unaltered.
As an example, let the memory location 3100 H and 3101 H contain 07 H and AF H respectively. Then execution of LHLD 3100 H copies the content of 3100 H into L and that of 3101 H into H register.

Register contents before instruction

H	XX	XX	L

Memory contents before instruction

3100	07
3101	AF

Register contents after instruction

H	AF	07	L
			H

Memory contents after instruction

3100	07
3101	AF

6. Memory location 3050 H is specified by HL pair and contains data FE H. Accumulator contains 14 H. Add the contents of memory location with accumulator. Store the result in 2050 H.

Ans. The program is written as follows:

LXIH, 3050 H ⇒ HL register pair points 3050 H memory location
ADD M ⇒ Add contents of 3050 H with accumulator
STA 2050 H ⇒ Result in accumulator is stored in 2050 H

Prior to execution of the program, memory location 3050 H is loaded with data FE H and accumulator with 14 H.

7. What the instruction DCR M stands for?

Ans. Here the memory location M is pointed to by the content of HL register pair. On executing DCR M, the content of the memory location is decremented by 01. Thus, if the memory location, as pointed to by HL register pair is AB H, then DCR M makes the content of that memory location to be AA H.

8. What the instruction DAD H stands for?

Ans. On executing the DAD H, the content of HL register pair is multiplied by 2. As an example if

H	04	20	L

Then DAD H makes the content of HL as follows:

H	08	40	L

9. Explain the two instructions (*a*) RAR and (*b*) RRC.

Ans. (*a*) RAR stands for 'rotate accumulator right through carry'.

Let the accumulator content = 95 H and carry flag = 0. Then execution of RAR will change the contents of CY flag and accumulator as follows:

	CY	D_7			ACC				D_0
Accumulator content before instruction	0	1	0	0	1	0	1	0	1
Accumulator content after instruction	1	0	1	0	0	1	0	1	0

Bit D_0 is placed in CY flag and the bit in the CY flag is placed in D_7.

(*b*) RRC stands for 'rotate accumulator right'.

Let, accumulator content = 95 H and carry flag = 0. Then execution of RRC will change the contents of CY flag and accumulator as follows:

	CY	D_7			ACC				D_0
Accumulator content before instruction	0	1	0	0	1	0	1	0	1
Accumulator content after instruction	1	1	1	0	0	1	0	1	0

Bit D_0 is placed in D_7 as well as in CY flag.

10. Explain the two instructions (*a*) RAL (*b*) RLC.

Ans. (*a*) RAL stands for 'rotate accumulator left through carry'.

Let accumulator content = 95 H and carry flag = 0. Then execution of RAL will change the contents of CY flag and accumulator as follows:

	CY	D_7			ACC				D_0
Accumulator content before instruction	0	1	0	0	1	0	1	0	1
Accumulator content after instruction	1	0	0	1	0	1	0	1	0

Here D_7 is placed in CY flag and the bit in CY flag placed in D_0

(*b*) RLC stands for 'rotate accumulator left'.

Let accumulator content = 95 H and carry flag = 0. Then execution of RLC will change the contents of CY flag and accumulator as follows:

	CY	D_7			ACC				D_0
Accumulator content before instruction	0	1	0	0	1	0	1	0	1
Accumulator content after instruction	1	0	0	1	0	1	0	1	1

Here D_7 is placed in D_0 as well as in CY flag.

11. Explain the instruction SPHL.

Ans. It stands for 'Copy H and L registers to the stack pointer'.

This instruction copies the contents of H and L registers into the stack pointer register. The content of H register goes to the high order address and the content of L register to the low order address of SP. H and L register contents remain unchanged.

Let H = 31 H and L = 32 H before the instruction is executed. Hence after its execution, it would be like as follows:

H	31	32	L	SP	31	32

12. Explain the instruction PCHL.

Ans. It stands for 'load program counter (PC) with HL contents'. The contents of H and L are copied into higher and lower order bytes of the PC. H and L register contents remain unchanged.

Let H = 31 H and L = 32 H before the instruction is executed. Hence after its execution, it would be as follows:

H	31	32	L	PC	31	32

13. Write down the result of EX–OR operation on accumulator and register B. Assume ACC = 18 H and B = 27 H.

Ans. The instruction used for this operation is XRA B.

ACC	=	0	0	0	1	1	0	0	0
B	=	0	0	1	0	0	1	1	1
EX–OR	⇒	0	0	1	1	1	1	1	1
	=	3F H							

The result 3F H of EX–OR is stored in accumulator.

14. Consider the following:

Contents of H, L and SP registers are A0 H, B2 H and 3062 H. Memory locations 3062 H and 3063 H contain 52 H and 16 H respectively. Indicate the contents of these registers after XTHL operation.

Ans. XTHL stands for exchange H and L with top of stack.

Different Register contents before XTHL instruction

H	A0	B2	L	3062	52
				3063	16
SP	3062				

After XTHL instruction is carried out, the different register contents would be

H	16	52	L	3062	B2
				3063	A0
SP	3062				

15. Bring out the difference between arithmetic shift and logical shift.

Ans. An example, with a right shift, will be considered to bring out the difference between arithmetic shift and logical shift operations.

Let the accumulator content is

	D_7							D_0
ACC =	1	1	0	0	0	1	0	0

Then arithmetic right shift operation will make the contents of ACC as follows:

	D_7							D_0
ACC =	1	1	1	0	0	0	1	0

i.e., the sign bit of the original number is retained at the MSB place.

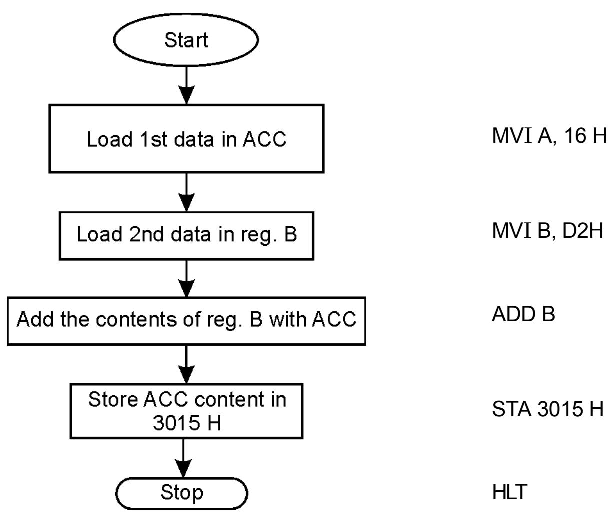
While logical right shift operation makes the ACC contents as follows:

	D_7							D_0
ACC =	0	1	1	0	0	0	1	0

i.e., the bits are shifted one bit position to the right.

16. Draw the flow chart and write down the program to add two numbers 16 H and D2 H. Store the result in memory location 3015 H.

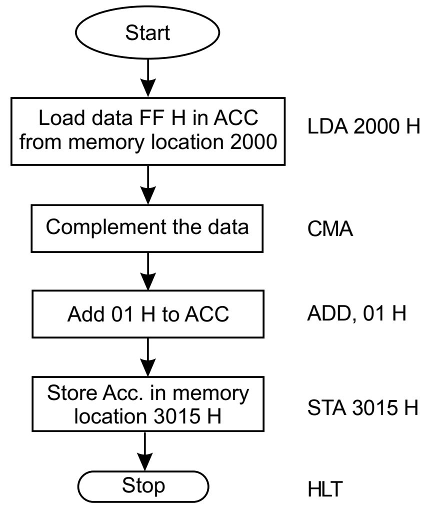
Ans. The flowchart for the above would look as follows.



17. Write a program, along with flowchart, to find the 2's complement of the number FF H, stored at memory location 2000 H. Store the result in memory location 3015 H.

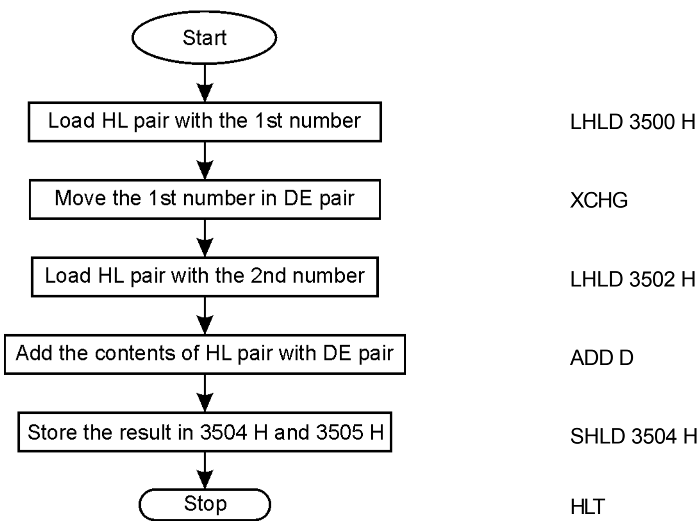
Ans. The flowchart, along with the program, is shown below:

Memory location 2000 H contains data FF H.



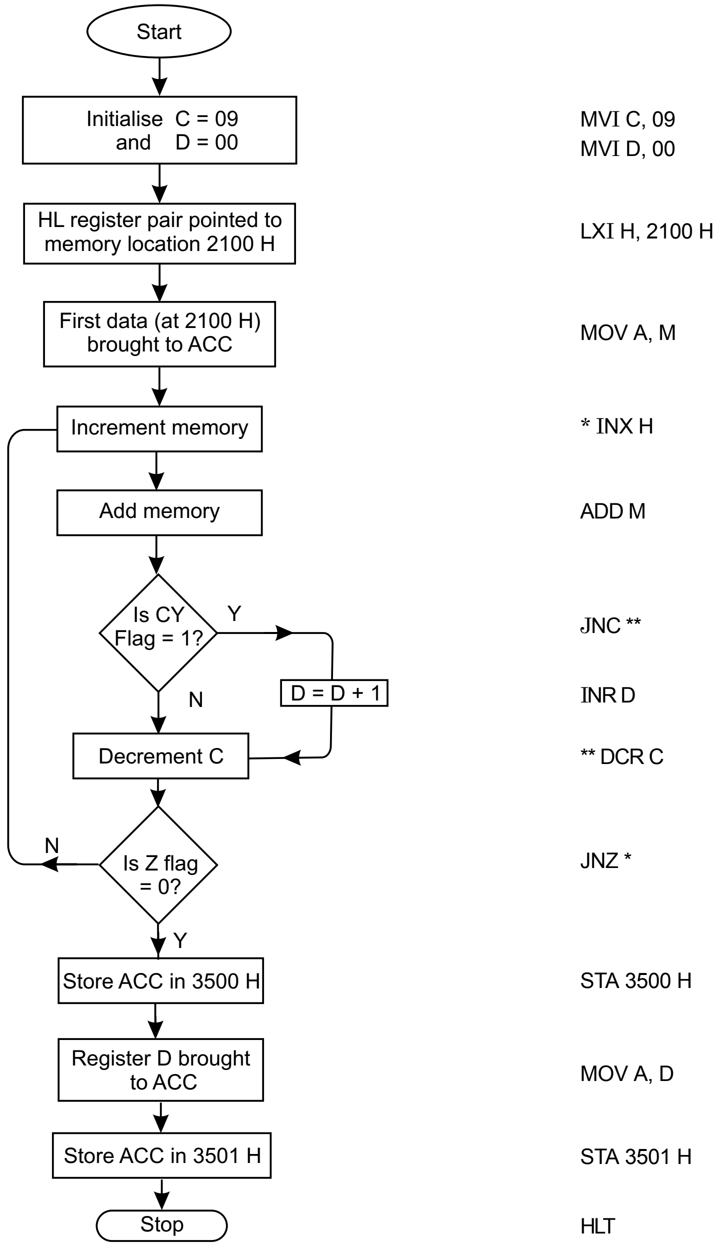
18. Write down the program to add two sixteen bit numbers. Draw the corresponding flowchart also.

Ans. Let the two sixteen bit numbers are stored at memory locations 3500 H to 3503 H. The result of addition is to be stored at memory locations 3504 H and 3505 H. The flowchart and the program will look like as follow:



19. Ten 8-bit numbers are stored starting from memory location 2100 H. Add the numbers, store the result at 3500 H memory location and carry at 3501 H. Draw the flowchart also.

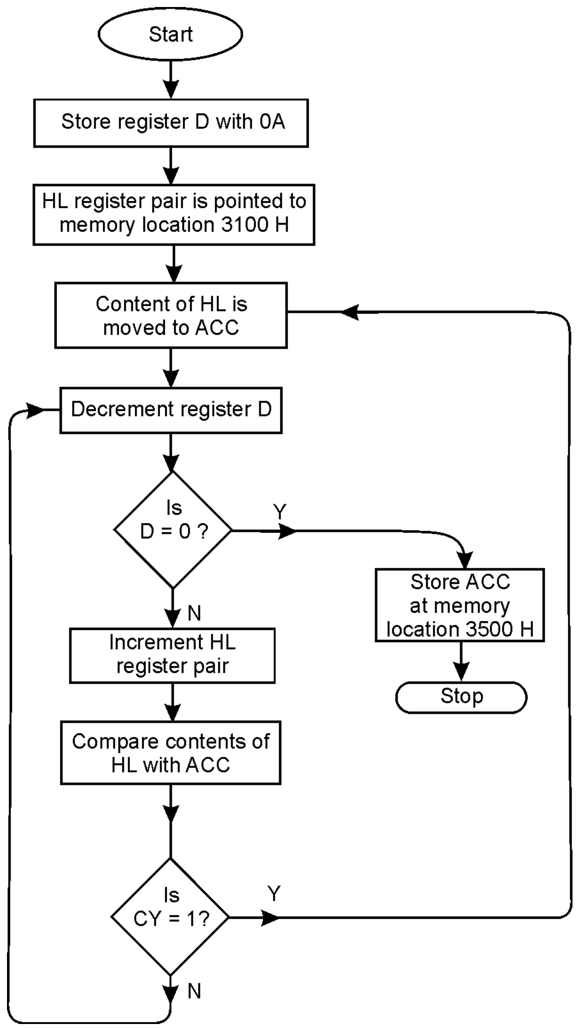
Ans. Initially, C register is stored with 09 H to take care of ten 8-bit data. Register D is initialised to 00 and stores the subsequent carry as addition is carried on. Final sum is stored at 3500 H while carry is stored at 3501 H.



The corresponding flowchart is also shown. Ten numbers of 8-bit data are stored starting from memory location 2100 H.

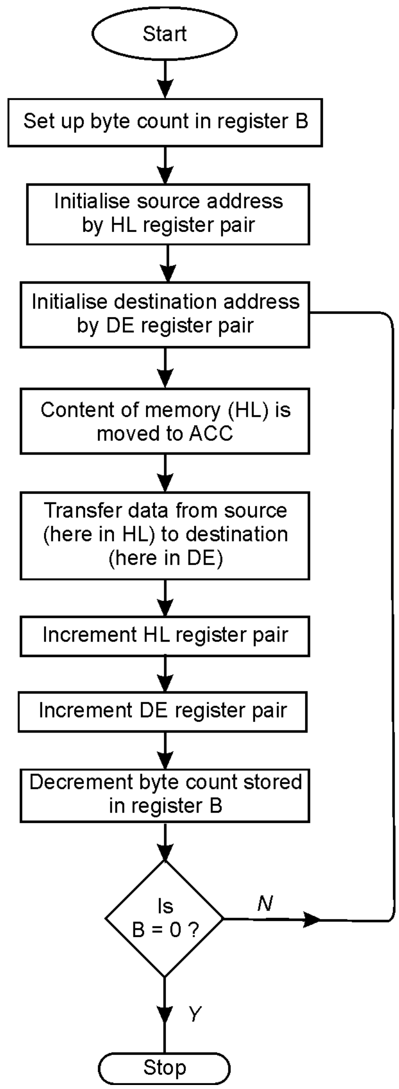
20. Ten 8-bit numbers are stored starting from memory location 3100 H. Find the greatest of the ten numbers and store it at memory location 3500 H.

Ans. Initially, D register is stored with 0A H—the number of data to be compared. The flowchart and the corresponding program are shown below:



21. Ten number 8-bit data are stored starting from memory location 2100 H. Transfer this entire block of data to memory location starting from 3100 H.

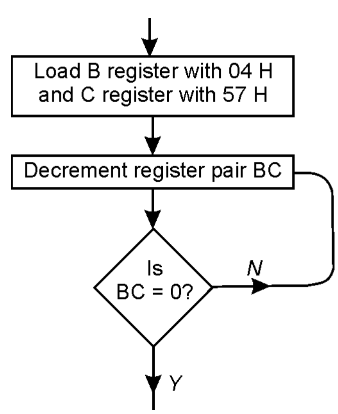
Ans. B register is used here as a counter which stores the number of data bytes to be transferred. The flowchart along with the program are shown below:




```
MVIB, 0A
LXIH, 2100 H
LXID, 3100 H
* MOV A, M
STAX D
INX H
INX D
DCR B
JNZ *
HLT
```

22. Set up a delay of 10 ms. Assume 3 MHz to be the microprocessor clock frequency.

Ans. Time delay is very easily generated by using a register (or a register pair—depending on the delay amount). Here the register pair BC is used to generate the delay with B loaded with 09 H and C with 03 H (calculations shown later). The flowchart and the program are shown side by side.




```
MVIB, 09 H
MVI C, 03 H
*DCX B
JNZ*
```

Calculations: A 3 MHz system clock corresponds to a time period of $0.\dot{3}$ µs. The instructions MVI B, MVI C, DCX B and JNZ take 7, 7, 6 and 7 T-states respectively, of which the two MVI instructions are outside the loop.

Time to execute MVI B and MVI C instructions

$$= (7 + 7) \times 0.\dot{3} \ \mu s$$

If the program loops n times, then time required for n loops

$$= \text{time for 1 loop} \times n$$
$$= (6 + 7) \times 0.\dot{3} \times n$$

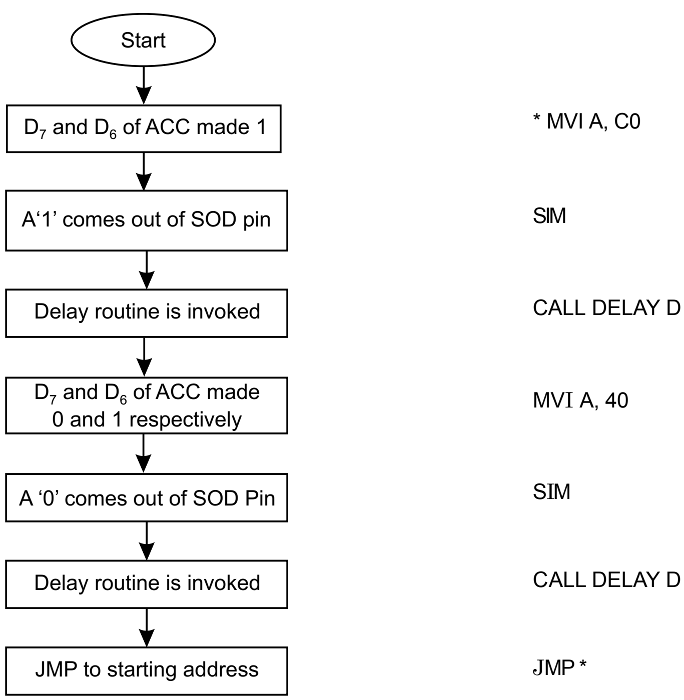
Thus $10 \times 10^3 = (7 + 7) \times 0.\dot{3} + (6 + 7) \times 0.\dot{3} \times n; \Rightarrow n \approx 2307)_{10} = 0903)_H$

23. Generate a square wave of 50% duty cycle through the SOD pin of 8085.

Ans. To generate a square wave, help of SIM instruction is taken. It reads like this

D_7	D_6						D_0
SOD	SOE	X	R7.5	MSE	M7.5′	M6.5′	M5.5′
Serial output data	Serial output data enable			Mask set enable			

Thus, for a ‘1’ to come out of SOD pin, D_7 and D_6 of accumulator to be made 1 while for a ‘0’ to be taken out of SOD pin, D_7 and D_6 of accumulator to be made 0 and 1 respectively. In both the above cases, D_5 to D_0 (six bits in all) are to be put to 0. Thus the program reads like this:

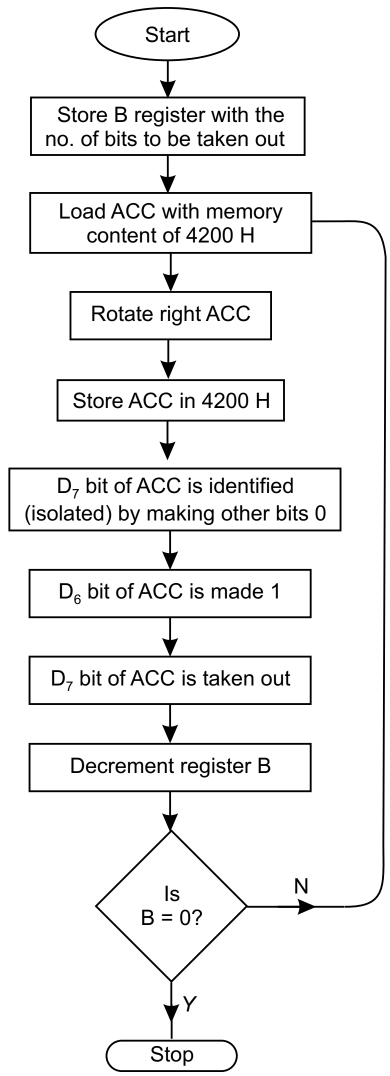


Thus the above generates a square wave which is obtained through SOD pin of microprocessor, having time period of 2D (2 × delay time).

If a non 50% duty cycle is desired, then the two delays are made different, as desired.

24. A data byte is stored in memory location 4200 H. This eight bit data is to be taken out of SOD pin, bit wise.

Ans. The flowchart and the program are as follows:




```
MVIB, 08
* LDA 4200
RRC
STA 4200
ANI 80
ADD 40
SIM
DCR B
JNZ *
HLT
```

6

Stack and Subroutines

1. What is a stack?

Ans. A stack is a group of memory locations in the R/W memory.

2. Why stack is used in a program?

Ans. The stack is used to store information temporarily during the execution of a program.

Also the stack is used in subroutine calls to store the return address.

As an example, data generated at a certain point in a program may be needed later in the program. This data is stored in the stack and retrieved when needed. Because the number of general purpose registers (GPRs) in a microprocessor is limited–hence not all the temporary data can be stored in them and this is where the stack plays its part.

3. How the stack is initialised?

Ans. The stack is initialised by a 16-bit register, called the stack pointer (SP) register.

4. Is initialisation of stack a must in a program?

Ans. No, it is not a must. If for programs for which any temporary data that are generated can be stored in GPRs and which don't require subroutine calls, there is no need to initialise the stack by the SP.

5. What the SP register does in a program?

Ans. The SP register keeps a track with regard to the storage and retrieval of data/information.

6. Who uses the stack?

Ans. The stack is used by both the programmer and the system.

Programmer uses the stack for storage/retrieval of data by using the PUSH/POP instructions respectively.

On the other hand, the system uses the stack to store return address whenever subroutine CALL is used.

7. Comment on the size of the stack.

Ans. It depends on the size of the R/W memory, as well as the program length. Since for a given system, the size of the R/W memory is fixed, thus smaller the size of the program, more would be the size of the stack.

8. Where, in the R/W memory, a programmer should initialise the stack?

Ans. The stack should be initialised at the high end of the memory map. This is explained in Fig. 6.1.

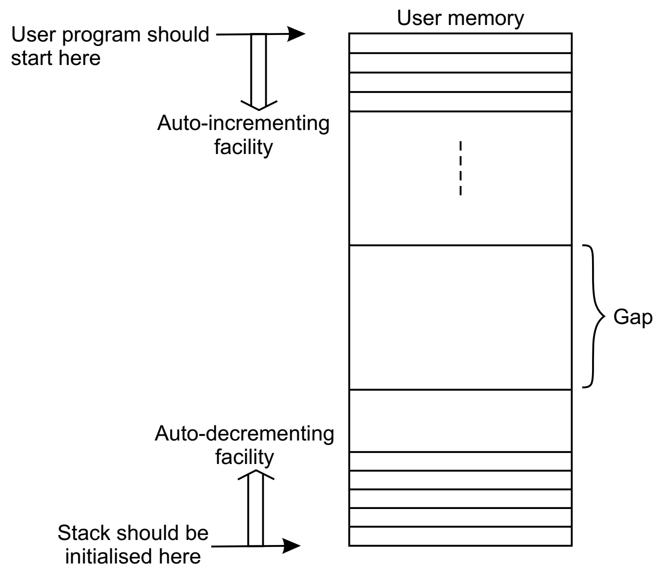


Fig. 6.1: Employing stack efficiently in a program

The user program should start at the low end of the memory map, while the stack should be initialised at the high end of the memory map. When the program is being executed, the PC is auto-incremented while as temporary data are stored in the stack, it is auto-decremented. Thus if sufficient gap is not maintained, the program area may get corrupted by the filling-in of the stack. In such a case, the program will 'crash'.

9. What type of memory is the stack?

Ans. Stack is a 'last-in first-out' or LIFO type of memory. This means that data which is pushed last into stack is popped out of it first.

10. How the stack is initialised?

Ans. The stack is initialised by means of the stack pointer. The software instruction is like this: LXI SP, 0044.

It means that the stack is initiated at the memory location 4400 by the stack pointer.

11. What are the software instructions related to stack operations?

Ans. The following are stack related software instructions:

LXI SP,	address
PUSH rp,	POP rp
PUSH PSW,	POP PSW
XTHL,	SPHL
CALL – RET	
DAD	SP

12. Does the stack have a fixed address?

Ans. No, the stack does not have a fixed address. But usually it is initialised at the top end of the memory address.

13. What are the typical errors associated with using stack in a program?

Ans. The possible errors in stack usage are: overflow or underflow of stack, PUSH or POP instructions not carried out in proper sequence, etc.

14. By how may memory locations SP is decremented/incremented when PUSH/POP instruction is performed.

Ans. SP is decremented/incremented by two memory locations when PUSH/POP is executed.

15. What is a subroutine?

Ans. A subroutine is a group of instructions, written separately from the main program, which performs a function that is required repeatedly in the main program.

16. Why a subroutine is used in a program?

Ans. Since a subroutine is called more than once by the main program, thus, use of subroutines saves precious memory space. The more the number of times a subroutine is called by the main program, the more is the saving of memory space.

17. Give some examples of subroutines.

Ans. Some examples of subroutines called by main program are: multiplication program, time delay subroutine, hexa-decimal converter subroutine, display subroutine, apart from any user specific subroutines that may be needed by a programmer.

18. Where do subroutines reside?

Ans. Standard subroutines like display program, time delay program, hexa-decimal conversion programs are supplied by the manufacturers of microcomputer in monitor ROM, while any user specific programs are written in the R/W memory.

19. How subroutines can be called from the main program?

Ans. Subroutines can be called from the main program either conditionally or unconditionally.

Mnemonic for unconditional call is CALL, while there are eight conditional calls. These are: CC, CNC, CZ, CNZ, CP, CM, CPE and CPO. CC stands for calling the subroutine if carry flag is set. CPE stands for calling the subroutine if parity flag is set (i.e., even parity).

20. How the program returns from the subroutine?

Ans. The return from the subroutine can be: conditional, unconditional.

The mnemonic for unconditional return is RET, while there are eight conditional return instructions viz. RC, RNC, RZ, RNZ, RP, RM, RPE and RPO. RZ stands for return if zero flag is set. RPO stands for return if parity flag is reset (odd parity).

21. Byte wise what are the lengths of CALL and RET instructions?

Ans. CALL is a 3-byte instruction, while RET is a 1-byte one.

22. Explain the CALL instruction.

Ans. CALL is a 3-byte unconditional instruction, upon the execution of which the main program branches to the starting address of the subroutine. The CALL instruction looks like this.

CALL memory address
(16-bit)

An example will clarify the situation.

The main program begins at 4080 by initialising the stack pointer at 621F. When CALL 4400 is encountered at 4100 in the main program, the return address following the CALL is 4103. The program counter is stored with the first address of the subroutine. Thus PC = 4400 and the content of the stack will look like as shown with the upper byte of the return address i.e., 41 stored at 621E and the lower byte (i.e., 03) at the stack address 621D. Thus, the program now starts at the subroutine starting address 4400.

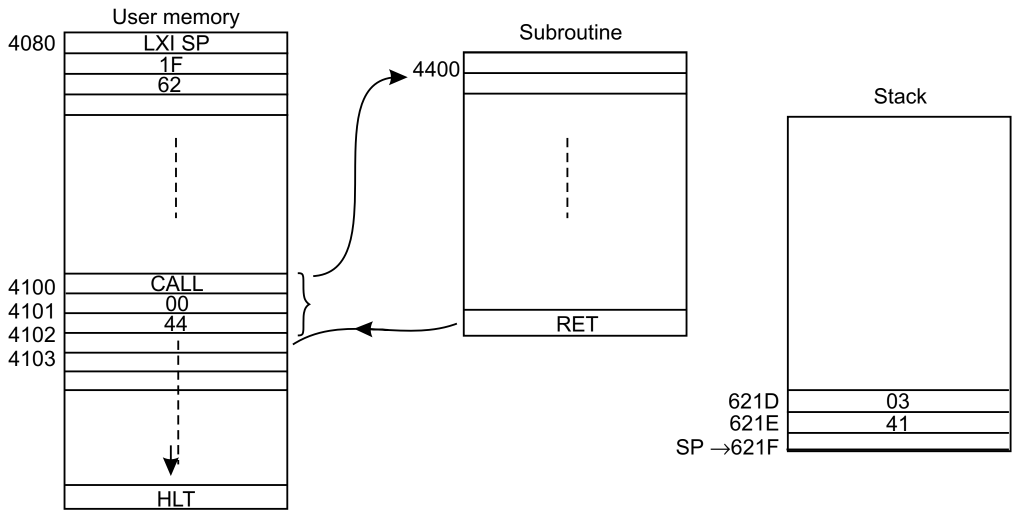


Fig. 6.2: Use of CALL-RET in a subroutine

When the last subroutine instruction RET is encountered, the two contents at the memory locations 621D and 621E are popped out so that the PC content now becomes 4103—which is the return address from the subroutine. Thus the program starts at 4103 in the main program and SP returns to 621F.

23. What is multiple calling of a subroutine?

Ans.

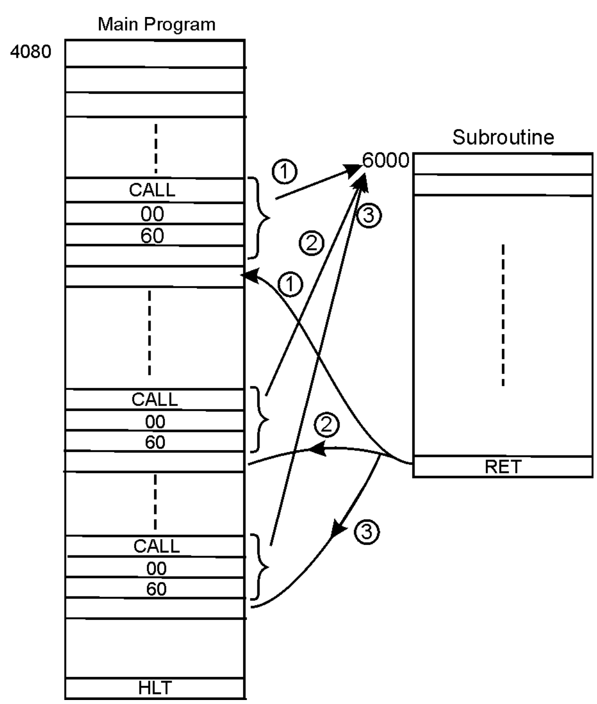


Fig. 6.3: Multiple calling of a subroutine

Subroutines are normally called more than once by the main program. Calling a subroutine more than once by the main program is called "multiple calling of a subroutine." This is shown in Fig. 6.3.

24. What is meant by 'parameter passing'?

Ans. Subroutines are scattered at many places in the memory and that they may be called from different locations in the main program. In such cases, various types of information/data are exchanged between the main program and the subroutine. This technique goes by the name "parameter passing".

25. Name the different types of subroutines.

Ans. The different types of subroutines are:

- multiple-calling of a subroutine
- nesting of subroutines
- multiple ending subroutines.

26. Explain nesting of subroutines.

Ans. The process of a subroutine calling a second subroutine and the second subroutine in its turn calling a third one and so on is called nesting of subroutines.

Theoretically speaking, the number of subroutines that can be called by this process is infinite but, in practice it is limited by the size of memory.

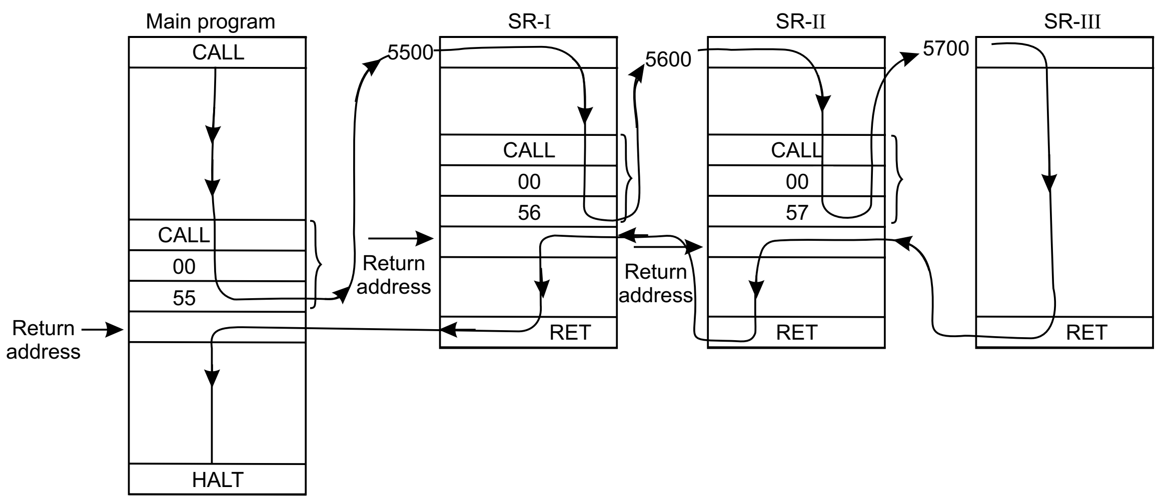


Fig. 6.4: Nesting of a subroutine

In Fig. 6.4, the main program calls the SR-I, SR-I in its turn calls SR-II while SR-II calls SR-III. The execution of the program is shown by the arrow heads. It is to be noted that each time a subroutine is called, the return addresses are saved automatically in the stack.

27. Draw an example of a multiple ending subroutine and explain.

Ans. The example of a multiple ending subroutine is shown in Fig. 6.5 which shows that the main program calls the subroutine starting at 5500.

Now the return from the subroutine can be via paths (1) or (2) (both conditional) or (3) (unconditional). Thus the return from the subroutine can be effected in three ways—

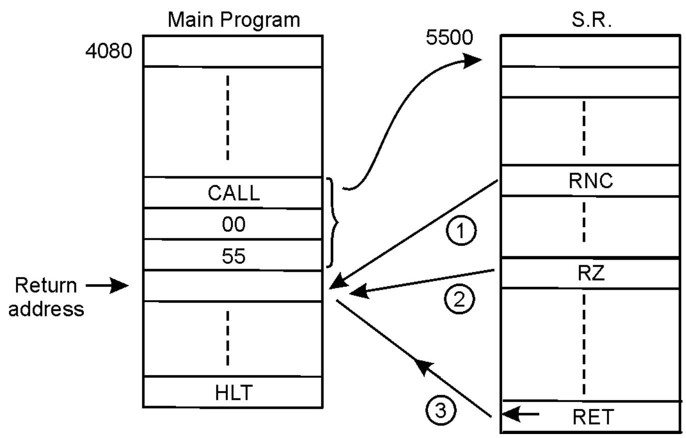


Fig. 6.5: Multiple ending subroutine

i.e., why it is called a multiple ending subroutine.

28. What are the techniques of ending a subroutine?

Ans. There are two methods of ending a subroutine—multiple ending and common ending. The example of each type follows:

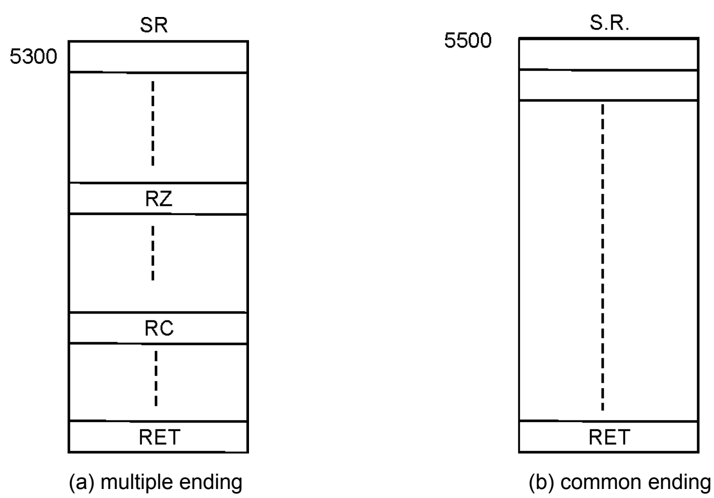


Fig. 6.6: Different ways of returning from a subroutine

29. For a 8085 based system, let the following two instructions are carried out

LXI SP, 0000 H

PUSH D

with D = 09 H and E = FA H assumed.

Show the stack contents after PUSH operation.

Ans. Stack gets decremented by one memory location from the one pointed to by the stack pointer. Since the pointer is at 0000 H, so on decrementing, SP now become FFFF H at which the content of D (i.e., 09 H) is stored. The SP is again decremented to become FFFE H where content of E (i.e., FA H) is stored.

	Register/Stack pointer before instruction			Stack content after instruction		Register/Stack pointer after instruction		
D	09	FA	E		D	09	FA	E
			FFFE	FA				
SP	0 0 0 0		FFFF	09	SP	F F F E		

30. Discuss recursive subroutine.

Ans. A recursive subroutine is a subroutine which is called by itself and are used with complex data structures, known as 'trees'.

For the flow diagram shown, if the subroutine is called with n = 4 (known as 'recursive depth'), then until n becomes 0 it will stay within the recursive subroutine.

Below is shown the flow diagram for a recursive subroutine.

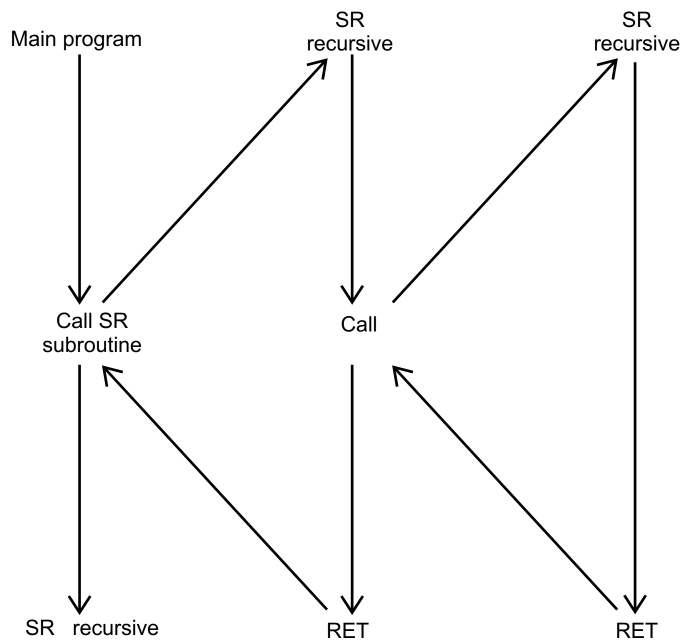


Fig. 6.7

If n ≠ 0

Decrement n

Call SR recursive

else

return.

31. In how many ways nested subroutines can be classified?

Ans. Nested subroutines can be classified in two ways as:

- Re-entrant subroutine
- Recursive subroutine

32. Discuss re-entrant subroutine.

Ans. In nested subroutines, many subroutines are there. In such a case, if a latter subroutine calls an earlier one, then it is known as re-entrant subroutine.

As an example, say a main program has two subroutines. The main program calls subroutine 1, then subroutine 1 calls subroutine 2. If now subroutine 1 is called from subroutine 2, then this falls under the category of re-entrant subroutines.

A scheme for such re-entrant subroutine is shown below:

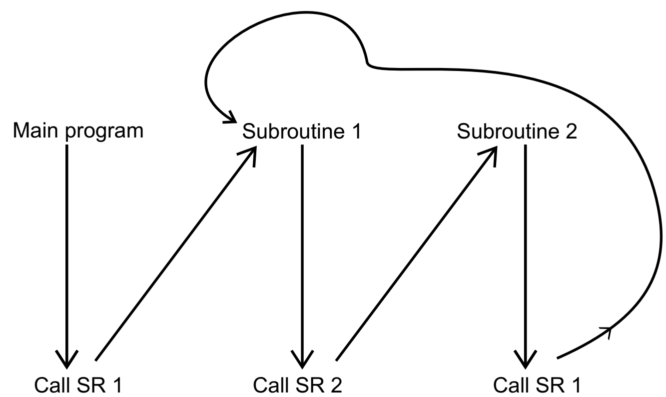


Fig. 6.8

33. How the stack is initialised?

Ans. There are two ways to initialise the stack

(*a*) Direct way

(*b*) Indirect way

Example of method (*a*) is LXI SP, 4400 H which loads the stack pointer with 4400 H. so that it points at the memory address 4400 H.

Example of method (*b*) is

LXI H, 16-bit data ⇒ load 16-bit data into HL register pair

SPHL ⇒ contents of HL register pair is loaded into SP.

In most of the cases the stack pointer is initialised by direct way, but method (*b*) is sometimes used when one wants to set the stack pointer by means of programming.

34. At what highest memory location can the SP be initialised.

Ans. The stack pointer is decremented by one memory location before data starts storing into the stack.

Hence, theoretically, the stack pointer can be initialised to a value which is one higher them the highest read/write memory location that is available.

35. Exchange the contents of DE register pair with that of HL register pair, using PUSH, POP instructions.

Ans. It is assumed that data are existing in the four registers—D, E, H and L. The following program then interchanges the contents of DE with that of HL and the stack is already initialised by the stack pointer.

PUSH D
PUSH H
POP D
POP H

36. Write a program which will store the contents of accumulator and flag register at 2000 H and 2001 H memory locations respectively.

Ans. The following is the program for the above, using PUSH-POP instructions.

LXISP, 4000 H:	Stack pointer is initialized at 4000 H.
PUSH PSW:	Accumulator and flag register contents pushed into stack
POP B:	Accumulator content goes to B and flag register content goes to C register.
MOV A, B:	Content of B taken to accumulator.
STA 2000 H:	Accumulator content stored into memory location 2000 H.
MOV A, C :	Content of C moved to accumulator.
STA 2001 H:	Accumulator content stored into memory location 2001 H
HLT:	Program halted.

37. What is meant by 'parameter passing technique' as used in subroutines?

Ans. Subroutines are programs, written separately from the main program, to process data or address variable that occurs repeatedly in the main program.

In order that the subroutine can process data, it is necessary to pass the data/address variable to the subroutine. This 'passing' of data/address variable is referred to as passing parameters to the subroutine.

There are four ways in which this 'passing' can be done, as mentioned below:

(*a*) by using registers

(*b*) by using pointers

(*c*) by using memory

(*d*) by using stack.

7

Data Transfer Techniques: Interfacing Memories and I/Os

1. Mention the two broad categories in which data transfer schemes are classified?

Ans. The data transfer schemes are broadly classified into two categories. These are
- Programmed data transfer
- Direct Memory Access (DMA) transfer.

2. Mention the types of programmed data transfer.

Ans. Programmed data transfer scheme is sub-divided into the following:
- Synchronous mode of data transfer,
- Asynchronous mode of data transfer and
- Interrupt driven mode of data transfer.

3. What are the features of programmed data transfer scheme?

Ans. In this scheme, data transfer takes place under the control of a program which resides in the main memory of the system. It is relatively slow and applied for cases when the number of bytes of data is small. This scheme is suitable for relatively slow peripherals.

4. Explain with the help of a block diagram, a typical programmed data transfer scheme.

Ans. The block schematic of a programmed data transfer scheme is shown below:

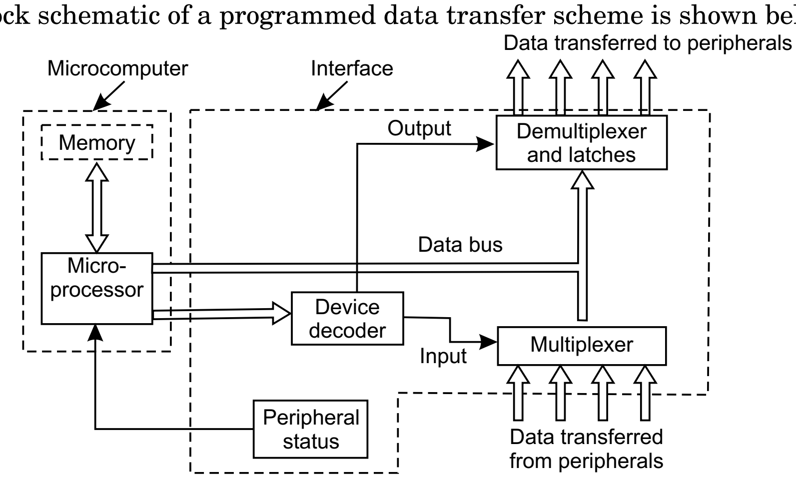


Fig. 7.1: Block diagram of a typical programmed-data transfer

For an input from or an output to a peripheral device, an I/O instruction is issued to a device decoder which therefore selects the corresponding multiplexer (for input) or demultiplexer (for output) respectively. The peripheral is tested for its readiness by means of a F/F. When the peripheral is ready, data from the peripheral goes to the accumulator of microprocessor or vice-versa for input or output peripherals respectively.

Since peripherals are usually slower than microprocessors, data are usually latched from the bus before actually handed over to the peripheral via the demultiplexer.

5. Mention and clarify the functions needed for peripheral interfacing.

Ans. The important functions are: buffering, address decoding, command decoding and timing and control.

Buffering is necessary to increase drive and also to synchronise data exchange between the microprocessor and peripheral.

A particular I/O is selected with the help of address decoding.

Command decoding is needed for some special I/Os that perform jobs other than data transfers—e.g. rewinding a tape drive.

For coordinating the above three, timing and control is needed.

6. In how many categories the interfacing peripherals are classified.

Ans. The interfacing peripherals are classified into two categories:

- General purpose peripherals.
- Special purpose peripherals.

7. Give some examples of general purpose and special purpose peripherals.

Ans. Examples of some general purpose peripherals are:

- Input/output ports
- Programmable Interrupt Controller (PIC)
- Programmable interval timer
- Programmable communication interface
- Programmable DMA interface
- Multipurpose programmable device.

While some of the special purpose peripherals are:

- Programmable CRT controller
- Programmable floppy disk controller
- Programmable keyboard and display interface.

8. Why it is relatively easy to interface memories than I/O devices? Elaborate.

Ans. It is relatively easy to interface a memory with a processor because memories are usually manufactured with the same technology as those of the CPUs and they are compatible to the CPUs with regard to speed and electrical compatibility.

But when an I/O device is interfaced with a processor, the following incompatibilities may arrive. These are:

- Speed incompatibility.
- Format incompatibility.
- Electrical characteristic incompatibility.

The first incompatibility arises because in many cases the I/O devices are slower than the processor so that a situation may arise when the processor is in a position to accept data but the peripheral, because of its slow nature, is unable to provide valid data.

The second incompatibility may arise if a 12-bit or 16-bit ADC or DAC is tried to be interfaced with an 8-bit microprocessor like 8085.

The third incompatibility may be due to current or voltage incomparability or both. Thus if a 12 V relay is to be driven by the SOD pin of 8085—both incompatibilities would be there. First the voltage (5 V) from the SOD pin is to be boosted to 12 V and also the current required to drive the relay should be provided. Thus it needs buffering.

9. Discuss the synchronous mode of data transfer.

Ans. Synchronous mode of data transfer is performed for peripherals whose timing characteristics is precisely known. In this mode the status of the device is not checked before undertaking any data transfer, that means, the device is assumed to be ready when data transfer takes place. This scheme is simplest amongst all the methods and minimum overhead in terms of hardware/software is needed to implement this scheme.

The following two figures show the flowchart for the data transfer scheme in synchronous mode. Figure 7.2 (a) corresponds to the case when the peripheral is in speed compatible with the CPU, while Fig. 7.2(b) corresponds to slower peripherals. For both Figs. 7.2(a) and (b), the timing characteristics of the peripheral should exactly be known. For the case of Fig. 7.2(b), the CPU sends a 'get ready' signal to the peripheral, followed by a wait for a predetermined time by the CPU. The CPU then executes the I/O instruction for data transfer to take place effectively.

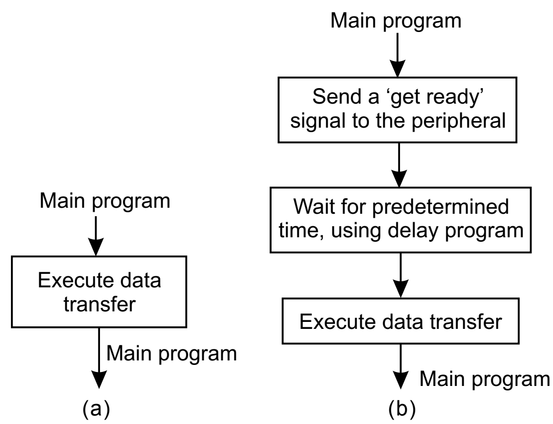


Fig. 7.2: Flowcharts for synchronous mode of data transfer (a) peripheral speed compatible to CPU, (b) peripheral slower than CPU

10. Discuss the asynchronous mode of data transfer scheme.

Ans. In this case the CPU initiates data transfer with the peripheral. The device (peripheral) status is checked by the CPU before undertaking data transfer. This mode is used when the timing characteristics of the device is unpredictable.

The flowchart for this scheme is shown in Fig. 7.3.

In this mode, the CPU confirms the readiness of the device status before undertaking data transfer. This is why this scheme is known by the name "handshaking I/O".

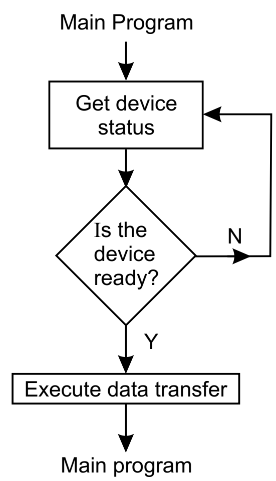


Fig. 7.3: Flowchart for asynchronous mode of data transfer

11. What is the main disadvantage of asynchronous mode of data transfer.

Ans. If there is an appreciable time gap from the instant the microprocessor starts checking the 'device ready' signal and its (device) actual readiness, the system loops the loop, as

is evident from Fig. 7.3. This is a time simply wasted by the processor until the device is ready with valid data. In an unfortunate situation, the system may enter into an infinite loop if the device does not become ready at all.

12. Discuss the interrupt driven mode of data transfer.

Ans. Main characteristic of this mode of data transfer is that data exchange between peripheral and the processor is initiated by the device.

This mode is used for data transfer with slow peripherals and also when the occurrence of data is unpredictable in nature.

The steps which are followed in this mode are:

- An interrupt is requested by a peripheral device.
- An acknowledgement of the request is issued by the processor at the end of the execution of the current instruction.
- The program then branches to Interrupt Service Subroutine (ISS) program at which the program corresponding to the interrupting device is already stored. The return address (in the PC) is stored in the stack along with other register contents as per program needs.
- Data transfer takes place under ISS.
- Interrupt system is enabled.
- The program then returns to the main program after loading the return address from stack in program counter (PC).

The flowchart corresponding to this scheme is shown below:

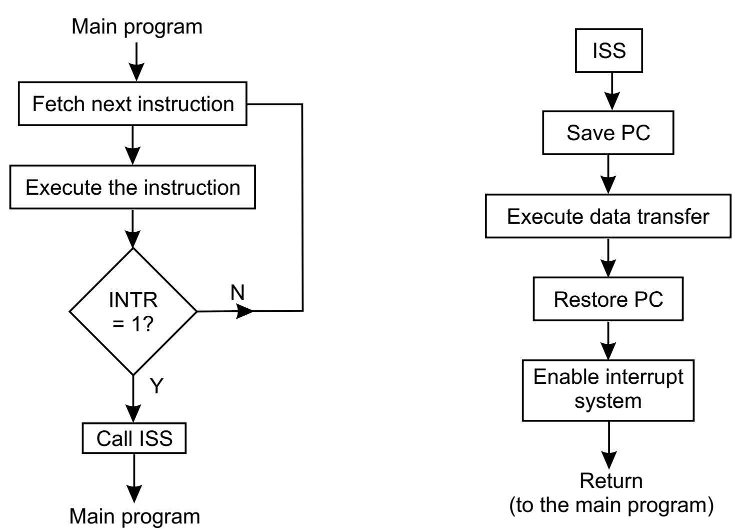


Fig. 7.4: Flowchart for the interrupt driven mode of data transfer

13. How does the asynchronous system differ from the interrupt driven mode of data transfer?

Ans. Whereas in the asynchronous mode of data transfer scheme it is the processor which goes on checking the device status, in the interrupt driven mode of data transfer scheme it is the device which interrupts the system.

14. In how many categories are interrupt driven mode of data transfer scheme divided?

Ans. It is divided into two categories: Polled interrupt and vectored interrupt. Again polled interrupt can be of two types: software polling, hardware polling.

15. Explain polled interrupt system.

Ans. Polled interrupt can be of two types and is used when many devices are connected to the system. In a polled interrupt scheme (whether hardware or software), each device is tested, using either hardware or software, until the device which has requested the interrupt, is identified. Corresponding to the device thus identified, the program is then diverted to the ISS written for that device.

16. Compare polled interrupt versus vectored interrupt.

Ans. In polled interrupt scheme (whether hardware/software) the priority of each device is fixed (by the programmer). It will take time before the interrupting device is identified. On the other hand, in the vectored interrupt scheme the requesting device causes the program to be branched to the ISS straightway. Hence vectored interrupt schemes are, in general, faster than polled interrupt schemes.

17. Describe the hardware polling scheme.

Ans. The schematic of a hardware polling scheme involving four devices is shown below. This scheme is also known as 'daisy-chaining'. The four device status flags from the four devices are ORed and taken to the INTR pin of the processor. A low $\overline{\text{INTA}}$ is issued by the processor at the end of the current instruction execution. The $\overline{\text{INTA}}$ signal is passed on to device 1—the highest priority device. If device 1 has interrupted the processor then it will identify itself with the data bus. If not, the $\overline{\text{INTA}}$ signal is passed on to device 2 and so on. Thus it is apparent that device 4 has got the lowest priority.

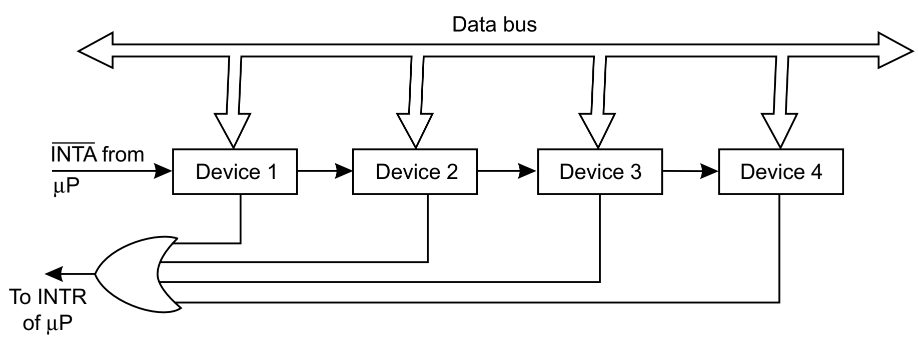


Fig. 7.5: Block diagram of hardware polling scheme (Daisy-chaining)

18. Describe the software polling scheme.

Ans. The flowchart for software polling scheme is shown below. It shows four devices whose status are checked in software one after the other. As per the scheme, device 1 has the highest priority while the lowest priority device is device 4. The status of each device are ORed and connected to INTR pin of the processor. On occurrence of an interrupt, the flag of each device is tested as per the software polling scheme.

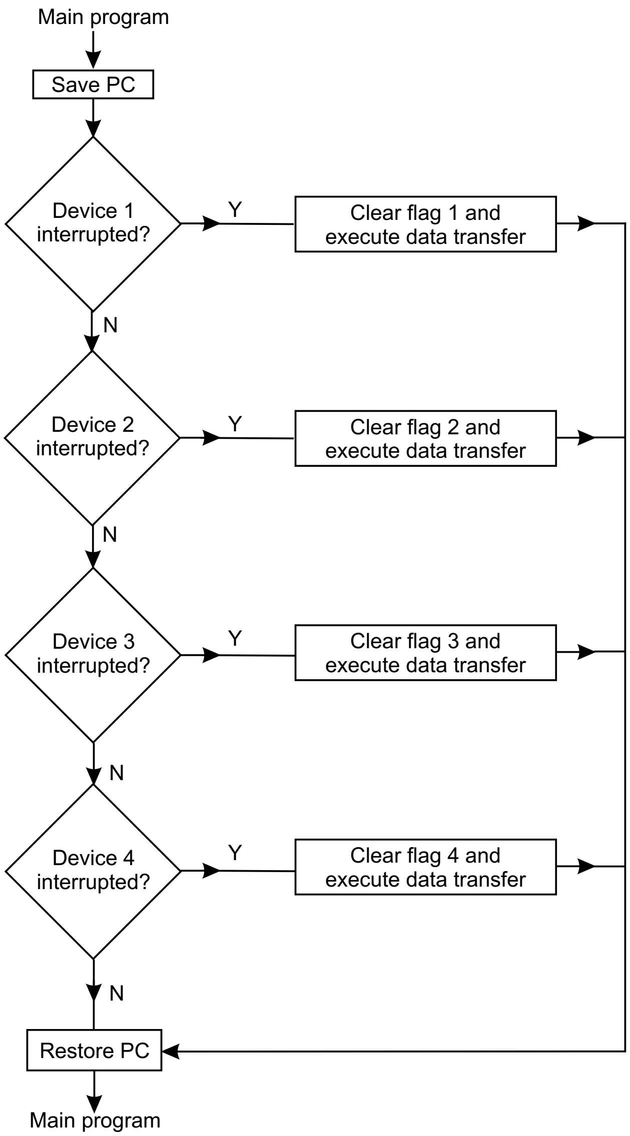


Fig. 7.6: Flow chart for software polling

19. Explain DMA (Direct Memory Access) mode of data transfer.

Ans. Instruction set of a processor provides for data transfer between processor registers and memory or I/O device. Thus when data transfer between memory and a I/O device is needed, it is done in two steps—from memory to accumulator of processor and then to I/O device or reverse. This slows down data transfer. DMA mode is introduced to overcome this.

In DMA mode, straight data exchange takes place between memory and I/O device bypassing the processor. This is done with the help of a DMA controller. In DMA mode, the DMA controller acts as a 'Master' and the processor as a 'Slave'.

20. What features must the processor and the DMA controller have to ensure proper operation in DMA mode?

Ans. The processor must have the following features to facilitate DMA mode of data transfer:

- An input line through which the processor accepts request from DMA controller for DMA mode of data transfer (This is the 'HOLD' pin for µP 8085).
- An output line through which the processor tells the DMA controller that it (processor) has accepted the request (This is the 'HLDA' pin for µP 8085)
- The processor must tristate its AB, DB and necessary control lines before handing over the control to the DMA controller.

The DMA controller IC must have the following features:

- An output line through which it requests the processor for DMA mode of data transfer.
- An input line through which it accepts the granted DMA request from the processor.
- Control over the AB, DB and the necessary control lines.

21. What is meant by address space?

Ans. It is defined as the set of all possible addresses that a microprocessor can generate.

22. What is meant by address space partitioning?

Ans. 8085 microprocessor has a 16-bit address bus so that it can address 2^{16} or 64 KB of address—called the address space of 8085. This total address space can be partitioned or allocated to memory or I/O devices so that they can be addressed properly. This is called address space partitioning.

23. What are the ways in which the address space can be partitioned?

Ans. The address space can be partitioned in two ways. These are:

- Memory mapped I/O scheme.
- I/O mapped I/O scheme.

24. Describe the memory mapped I/O scheme.

Ans. In this scheme, there is only one address space. This address space is allocated to both memory and I/O devices. Some addresses are assigned to memories and some to I/O devices. The address for I/O devices is different from the addresses which have been assigned to memories. An I/O device is also treated as a memory location. In this scheme one address is assigned to each memory location and one address is assigned to each I/O device.

In this scheme, all data transfer instructions of the microprocessor can be used for transferring data from and to either memory or I/O devices. For example, MOV D,M instruction would transfer one byte of data from a memory location or an input device to the register D, depending on whether the address in the H-L register pair is assigned to a memory location or to an input device. If H-L contains address of a memory location, data will be transferred from that memory location to register D, while if H-L pair contains the address of an input device, data will be transferred from that input device to register D.

This scheme is suitable for small systems. In this scheme, $IO/\overline{M}$ signal is not used to distinguish between memory and I/O devices. An I/O device is interfaced in the same manner as a memory device.

25. Explain the I/O mapped I/O scheme.

Ans. Some CPUs provide one or more control lines (for example, $IO/\overline{M}$ line for 8085), the status of which indicates either memory or I/O operation. When the status of $IO/\overline{M}$ line

is high, it indicates I/O operation and when low, it points to memory operation. Thus, in this case, the same address may be assigned to both memory or an I/O device—depending on the status of IO/$\overline{M}$ line.

The above scheme is referred to as I/O mapped I/O scheme. Here two separate address spaces exist—one space is meant exclusively for memory operations and the other for I/O operations. Usually, the space earmarked for I/O is much smaller than memory space.

26. Pictorially show the memory mapped I/O and I/O mapped I/O scheme.

Ans. The following figure shows, pictorially, both the schemes. Here it is assumed that the system has a 64 KB of memory and 256 I/O space.

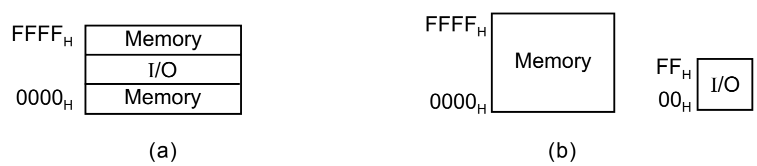


Fig. 7.7: (a) Address space for memory mapped I/O scheme (b) Address space for I/O mapped I/O scheme

27. What is another name of I/O mapped I/O and how many I/O's can be accessed in this mode?

Ans. I/O mapped I/O is also known as standard I/O. A maximum of 2^8 = 256 I/Os can be addressed in this mode, because in this mode a 1-byte address is specified.

28. Bring out the distinguishing features between memory mapped I/O scheme and I/O mapped I/O scheme.

Ans. The following table shows the distinguishing features of the two schemes:

Table 7.1: Comparison between Memory Mapped I/O and I/O Mapped I/O

S. No.	Memory Mapped I/O		I/O Mapped I/O
1.	Address width is 16-bit. A_0 to A_{15} are used to generate address of the device.	1.	Address width is 8-bit. A_0 to A_{15} lines are used to generate address of the device.
2.	$\overline{MEMR}$ and $\overline{MEMW}$ control signals are used to control read and write I/O operations respectively.	2.	$\overline{IOR}$ and $\overline{IOW}$ control signals are used to control read and write I/O operations respectively.
3.	Instructions available are STA addr, LDA addr, LDAX rp, STAX rp, ADD M, CMP M, MOV r, M, etc.	3.	IN and OUT are the only available instructions.
4.	Data transfer takes place between any register and I/O device.	4.	Data transfer takes place between accumulator and I/O device.
5.	Maximum number of I/O devices that can be addressed is 65536 (theoretically).	5.	Maximum number of I/O devices that can be addressed is 256.
6.	Execution speed using STA addr, LDA addr is 13 T-state and for MOV M, r, etc., it is 7-T states.	6.	Execution speed is 10 T-states.
7.	Decoding 16-bit address will require more hardware circuitry.	7.	Decoding 8-bit address will require less hardware circuitry.

29. What are the instructions available in memory mapped I/O and I/O mapped I/O scheme?

Ans. The instructions available in memory mapped I/O scheme are LDA, LDAX, STA, STAX, MOV M, r etc. while those for I/O mapped I/O scheme are IN and OUT.

The CPU sends out an 8-bit code to identify the particular port address.

30. Mention the most important advantage of memory mapped I/O scheme over I/O mapped I/O scheme.

Ans. The main advantage of memory mapped I/O scheme is that all memory reference instructions are available in this scheme.

31. 2 KB RAM, 2 KB ROM, one input and one output device are to be interfaced with 8085 microprocessor. Employ memory mapped I/O scheme to execute the above.

Ans. Total memory capacity: 2 KB RAM and 2 KB ROM = 4 KB. This requires 12 address lines (2^{12} = 4 K). Out of 12, 11 address lines (A_0 to A_{10}) of μP are connected to 11 memory address lines. Address line A_{11} is used as chip select. The RAM chip is selected when A_{11} is low and ROM gets selected for A_{11} = 1.

The two address lines are connected to A_{15} and A_{14}. Thus, the 16-bit address line would be:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
I_1	I_2	X	X	C	M	M	M	M	M	M	M	M	M	M	M

in which

I represents I/O devices
C represents chip select
M represents memory
X represents don't care.

Status of A_{15}, A_{14} and A_{11} selects either memory or I/O devices as follows:

A_{15}	A_{14}	A_{11}	*Selected device*
0	0	0	RAM
0	0	1	ROM
1	0	X	Output device
0	1	X	Input device.

32. Employ I/O mapped I/O scheme for Q 31.

Ans. In this scheme, $IO/\overline{M}$ signal distinguishes between an I/O and a memory.

Memory addressing is done exactly in the same manner as discussed in Q 31.

In this scheme, the input and output devices are identified by $\overline{IO/\overline{M}}$ along with A_0 address line, while the RAM and ROM are identified by $IO/\overline{M}$ signal along with A_{11} address line. The scheme of decoding is shown in Fig. 7.8.

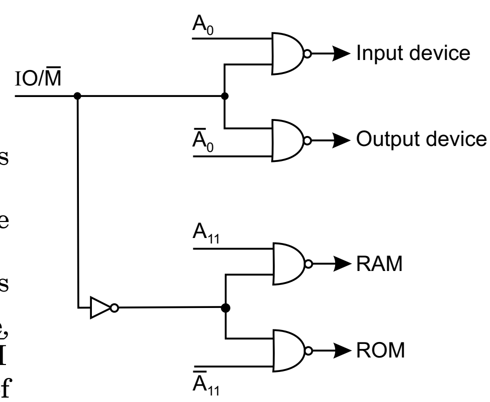


Fig. 7.8: Decoding circuit to distinguish memory from I/O devices

33. Why WAIT states are used in μC based systems?

Ans. WAIT states are used to interface slow peripherals to the processor.

8

Memory

1. What is a memory?

Ans. A memory is a device that stores information in electrical, magnetic or optical form. A μC based system, which operates on digital logic, holds binary information. Semiconductor memories are used in μC based system.

2. Why semiconductor memories are used as main memories in μC based systems?

Ans. Semiconductor memories have become very popular and widely used because of their high reliability, low cost, high speed and ease with which memory size can be expanded.

3. What is the use of memory in a μC based system?

Ans. Memories are used for storage of both program and data.

4. In how many categories memories can be classified? Give examples and distinguish between them.

Ans. It can be categorised into two ways:

- Primary memory or main memory or working memory.
- Secondary memory or auxiliary memory or mass storage.

RAM and ROM comprise the primary memory while magnetic tapes, magnetic disks, floppy disks or compact disks (CDs) are examples of secondary memory.

Distinction between the two types of memories are:

Table 8.1: Comparison between primary and secondary memory

S. No.	Primary memory		Secondary memory
1.	Can store less amount of data.	1.	Can store huge amount of data.
2.	Faster speed of operation.	2.	Slower speed of operation.
3.	Can be volatile/non-volatile in nature.	3.	Always non-volatile in nature.
4.	Program/data used by the programmer are resident in the main memory.	4.	Not used for such purpose.
5.	Can be directly accessed by CPU.	5.	Cannot be directly accessed by CPU but can be accessed through I/O ports or in a serial format using hardware/software.
6.	If the CPU has n address lines, a maximum of 2^n main memory locations can be accessed.	6.	No such relation exists.
7.	Less costly.	7.	Costlier than main memory.

5. What is the basic memory operation?

Ans. 'Read' and 'Write' are the basic operations performed by a memory device. The process of storing data into memory is called 'Writing' and retrieving data/instruction from the memory is called 'Reading'. Each memory location is identified by a particular address corresponding to the memory location. For reading or writing data, the particular memory location is to be identified by its address first and then only reading/writing is done.

6. What is a cell?

Ans. Memories are made of storage elements, which can store one bit of data. Each storage element is called a cell. In semiconductor memories, flip-flops act as storage elements.

7. What is meant by 'working length' of a memory?

Ans. At each memory location, which is identified by its address, one or more number of bits can be stored. The number of bits that a particular memory location can store is known as working length of a memory. This also goes by the name of word size.

8. What is meant by "access time" of a memory? On which does it depend?

Ans. It is the time required to perform a read operation. Putting another way, it is the time between the memory receiving a new address and putting out the memory content on its output. This time is symbolised by t_{acc}.

9. What is a sequential access memory (SAM)?

Ans. It is a kind of memory for which t_{acc} is not constant for all memory locations/positions. An example of a SAM is a magnetic tape backup. The latter the data is stored, the more the t_{acc} time for such a data. SAM is used only when data retrieval is always sequential in nature.

10. What is a volatile memory?

Ans. It is such a memory whose stored information is lost when electrical power is removed.

Semiconductor memories may be volatile/non-volatile in nature, while magnetic memories are non-volatile.

11. A semiconductor memory is specified as 4 K × 8. Indicate the number of words, word size and total capacity of this memory.

Ans. Total number of words or memory locations that can be accessed is

$$= 4\text{ K} = 4 \times 1024 = 4096$$

$$\text{Word size} = 8$$

and total capacity of the memory is

$$= 4096 \times 8 = 32768$$
$$= 32,768 \text{ bits.}$$

12. Diagrammatically show a 1 K × 8 memory, indicating the relevant pins.

Ans. The following figure shows the diagrammatic representation of a 1 K × 8 memory.

The memory will be selected only if $\overline{CS}$ (chip select) signal is low. 1 K represents 1024 memory address locations so that 10 number of address inputs ($A_0 - A_9$) are required. $I_0 - I_7$ and $O_0 - O_7$ (both 8 pins) represent respectively data inputs and outputs.

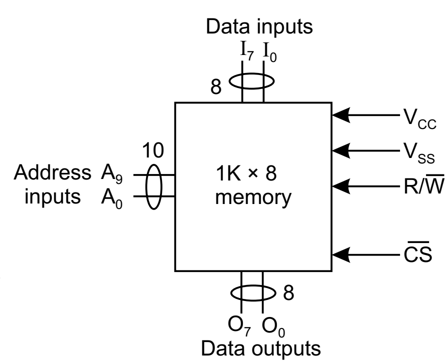


Fig. 8.1: A 1 K × 8 memory module

13. Show the different versions of semiconductor memories available.

Ans. Basically semiconductor memories can be RAM (Random Access Memory) or ROM (Read Only Memory)—both of which are available in bipolar technology or MOS (Metal Oxide Semiconductor) versions. The different versions available are:

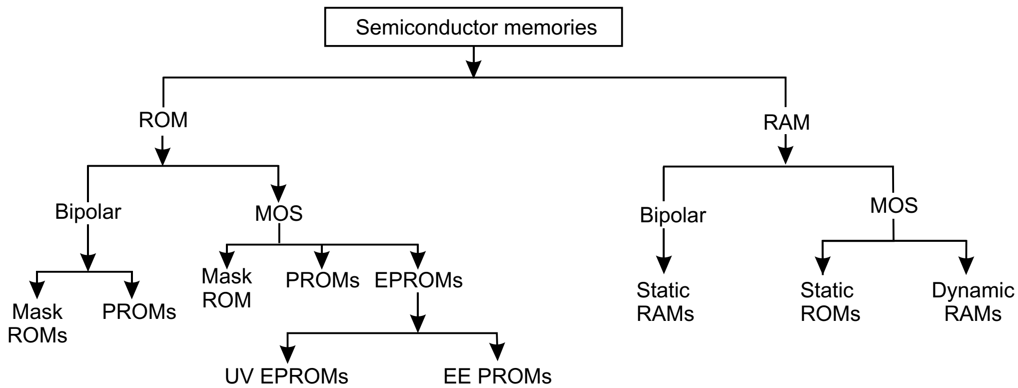


Fig. 8.2: Semiconductor memory hierarchy

14. What are the characteristic features of a RAM?

Ans. The essential features of a RAM are:

1. It is volatile in nature—i.e., when the power is switched off, data stored at the different memory locations are totally lost.
2. When data/instruction is written at a memory location, previous data stored at this location is destroyed and replaced by the new data.
3. When data/instruction is read from an address location, the existing data is not lost (destroyed).

15. What is a CAM?

Ans. It stands for Content Addressable Memory. It is a special purpose RAM which performs association operation, in addition to read/write operations.

16. What is a PLA? What are the advantages of using a PLA?

Ans. A PLA stands for 'Programmable Logic Array'. It is a non-volatile random access storage device. It can be used for implementing random logic circuitry and ROMs.

A PLA consists of two arrays (AND and OR), an input buffer and an output buffer.

The main advantages of using a PLA are: 1. Fastest access time 2. High switching speed 3. Compact circuitry 4. Implementing random logic circuitry.

The block diagram of a PLA is as follows:

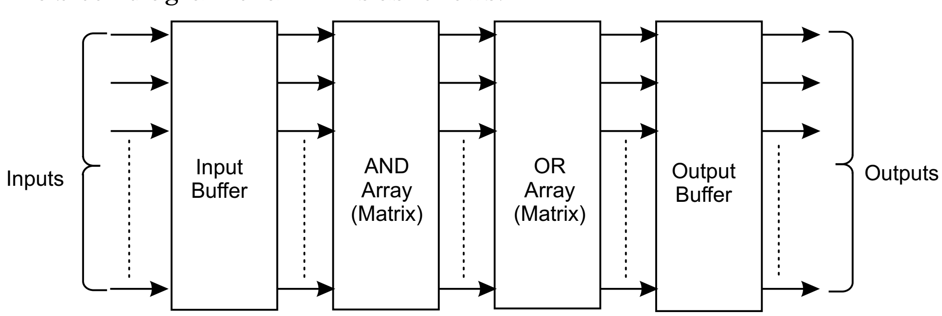


Fig. 8.3: Block diagram of PLA

17. Applicationwise, write down the differences between RAM and ROM.

Ans. RAMs are used for writing/development of a program while ROMs are used for such programs which are repetitively required in a program—for example hexa-decimal conversion table, display program, time delay program and most importantly programmed instructions for system initialisation and operation.

18. What are the incompatibilities that may arise when memory devices are interfaced with μP? How are they removed?

Ans. The incompatibilities are: electrical, speed and bus.

Electrical incompatibility is removed by using bidirectional and unidirectional bus drivers.

Speed incompatibility is overcome by either slowing down the CPU clock or by inserting wait states.

Bus incompatibility arises because of multiplexed address-cum-data bus, Read-Write signals differing for CPU and memory. The bus incompatibility can be removed by using extra hardware.

19. What are bit and byte organised memories?

Ans. For a $2^n \times m$ memory, where n is the number of address lines such that a maximum of 2^n memory locations can be accessed and m is the word length, if m is 1 then the memory is said to be bit organised, whereas if m is 8, then the memory is said to be byte organised.

20. Mention the essential features of Mask programmable ROM and PROM.

Ans. The essential features of these two are tabulated as follows:

Table 8.2: Comparison between mask programmable ROM and PROM

S. No.	Mask programmable ROM		PROM
1.	Manufactured by making special masks.	1.	Manufactured by blowing fusible nichrome wire links.
2.	Programmed at the factory premises.	2.	Programmed by the user.
3.	No reprogramming possible.	3.	No reprogramming possible.
4.	Less costly.	4.	More costly.

21. What is a EPROM? Mention its two types and compare.

Ans. EPROMs are manufactured with NMOSFET technology with an isolated gate structure. There are two types of EPROM. These are:

UVEPROM : Ultraviolet erasable programmable Read-only-Memory
EEPROM (also known as EAPROM) : Electrically erasable programmable Read-only-Memory

A comparison between UVPROM and EEPROM is made hereunder:

Table 8.3: Comparison between EEPROM and UVPROM

S. No.	EEPROM		UVPROM
1.	Can be erased and programmed with electrical signals.	1.	Can be erased and programmed with ultraviolet light.
2.	The voltage on the floating gate structure allows storage of information.	2.	The photo current from the insulated gate structure allows storage of information.
3.	Higher speed of operation.	3.	Lower speed of operation.
4.	More expensive.	4.	Less expensive.
5.	Relatively easy to manufacture.	5.	Relatively difficult to manufacture.
6.	Erasing takes several minutes.	6.	Erasing takes several minutes.
7.	In-situ erasing possible.	7.	In-situ erasing not possible.
8.	Byte wise erasing possible.	8.	Erasing wipes out the entire memory.
9.	Less packing density.	9.	More packing density.

22. Name some of the application areas of ROM.

Ans. ROM has applications in many areas. Some of these are: bootstrap memory, firmware, data converters, function generators, data tables, etc.

23. What is a firmware?

Ans. A set of program/data must be made available to a microprocessor based system whenever it is powered on. This program/data is stored in ROM in what is called a firmware.

Laptop computers, PCs, etc. store their operating system programs and language interpreters (like PASCAL, BASIC, etc.) in ROM firmware.

24. What is a boot strap memory?

Ans. Most large computers store their operating system in external mass memory like magnetic disk. For such computers, a small program, called the boot strap program, is stored in the ROM area of the computer. On powering such a computer, the bootstrap program is first executed which then loads the operating system programs from magnetic disk into the main (internal) memory of the computer. The OS is then run by the computer so that it becomes ready to accept user commands, this 'start-up' process is called 'booting of the system'.

25. What is a flash memory?

Ans. This is an alternative to EPROMs and EEPROMs. It provides for very fast read access, in-circuit erasability. Its density almost matches that of EPROM, at the same time it is cost effective also. They consume less power and are able to withstand severe shock and vibration.

Most flash memory chips erase all the cells in the chip, although some newer version of flash memory chips has sectorwise erase option, i.e., 512 bytes of information can be erased at a time.

Typical values for memory write (per byte) are 10 μs, 100 μs and 5 ms for flash type, EPROM and EEPROM, respectively.

Flash memories are used in digital cellular phones, LAN switches, embedded controllers, digital set-up boxes, etc.

26. What are the different types of SAMs?

Ans. SAM stands for Sequential Access Memory. Its different versions are: Shift Memory, Charged Coupled Devices (CCDs) and Bubble memories.

27. What kind of memory device is a CCD?

Ans. It is volatile type memory device, whose contents can be accessed in a serial manner.

28. Name the basic storage element in a CCD.

Ans. The basic storage element in a CCD is a MOS capacitor.

29. Mention the operations involved in a CCD memory.

Ans. The main operations involved are:

- Converting a digital input signal into charge.
- Transferring the charges in a sequential manner.
- Charges at the outputs are converted back into digital form.

30. Discuss Bubble Memory.

Ans. Magnetic bubble memory is a solid state device. It is highly reliable, small in size, light in weight and its power consumption is low. Its access time is high—i.e., it is a slow device. A typical figure is 100 k bits/second.

In this type of memory, data is stored in magnetic bubbles. This is done in a thin film of magnetic material. A '1' is represented by the presence of a bubble while the absence signifies a logical '0'.

It is a non-volatile semi random access type memory. Its readout is non-destructive in nature.

A bubble memory contains several loops, each loop of which contains a large number of bits. For read or write operation to be done, each loop possesses a 1-bit viewing window.

INTEL 7110 is a bubble memory chip having 1 M bits storage capacity while INTEL 7114 has a 4M bits capacity.

31. Explain how the entire addressable memory space of 8085 can be conceived of the pages of a book?

Ans. The entire addressable memory space is 2^{16} = 65,536 lines, because 8085 has 16 address lines. This is arbitrarily divided into 256 pages (0 to 255 pages) with each page consisting of 256 lines (0 to 255 lines), as shown in Fig. 8.4

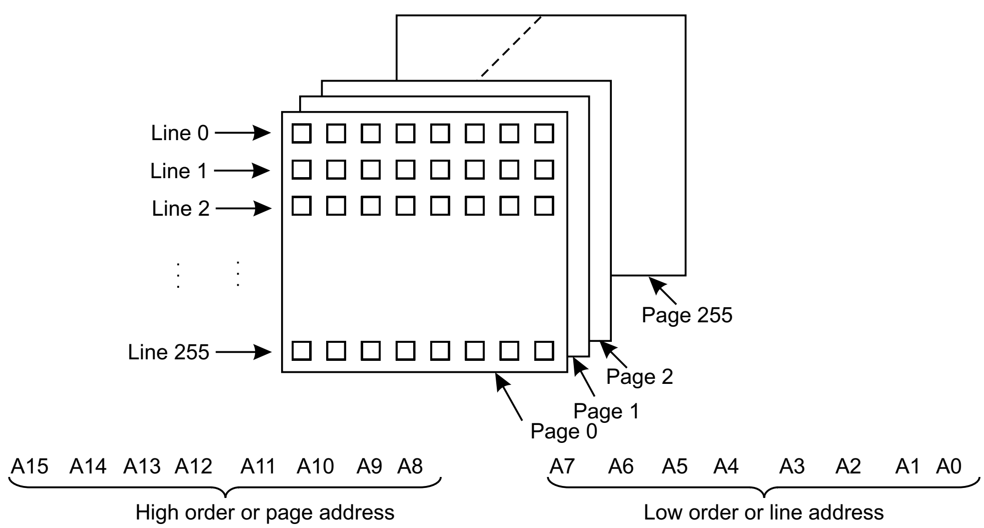


Fig. 8.4: Arranging total memory space of 8085 into pages and lines

The sixteen address lines can be divided into higher ($A_{15} - A_8$) and lower ($A_7 - A_0$) eight bits. On each page there are 256 lines defined by the lower byte ($A_7 - A_0$), while each page of the book is defined by the different combinations of the upper byte ($A_{15} - A_8$).

The following figure shows an arbitrary but convenient way of showing the total memory space, with each block consisting of 4 K lines i.e., 16 pages.

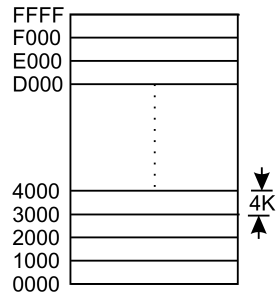


Fig. 8.5: Total memory space of 8085

32. Mention the memory capacities corresponding to the number of address lines 10, 11, 12 .... 16.

Ans. The memory capacities corresponding to the number of address lines 10, 11, ... 15, 16 are 1 K, 2 K, 4 K, 8 K, 16 K, 32 K and 64 K, respectively.

33. What is a dual-port RAM? Mention its use.

Ans. It is a RAM having separate input and output pins. Because separate input and output pins are used (unlike normal RAM, in which data bus is bidirectional), it is used in high speed applications like the video RAM in a PC. In such a case the system bus feeds the RAM input with new informations while the video card reads from the RAM to constantly refresh the screen.

34. What is a NVRAM? Mention its use.

Ans. It stands for non-volatile RAM and is made by the combination of static RAM and an EEPROM. The memory structure of static RAM and EEPROM are mirror images of each other. When powered, NVRAM behaves as a normal RAM. When there is a power failure, RAM data is saved in EEPROM in less than 4 ms. On restoration of power, the reverse process occurs with contents of EEPROM sent back to RAM. Two pins $\overline{\text{STORE}}$ and $\overline{\text{RECALL}}$ of the NVRAM are used to store data in EEPROM and transfer it back to RAM respectively, both these transfers occurring on 'low' on the two pins.

The advantage of using an NVRAM is that a battery backup is not required to save the RAM data in case of power failure.

The NVRAM finds its use in information storage such as time, date, monitor setting and other computer configurations.

35. What is a memory map?

Ans. A microcomputer system uses a mix of ROM, RAM for its addressable memory space. Some of the memory space may be left unimplemented or open.

The memory map is a guide showing how the entire system memory has been allocated to ROM, RAM so that any future memory expansions can be executed effectively.

36. Draw a static RAM (SRAM) cell and explain its operation.

Ans. A standard SRAM consists of six transistors connected to form a R-S flip-flop. A SRAM cell is shown below. The transistors Q_1 and Q_2 form the cross-coupling transistors required for latching.

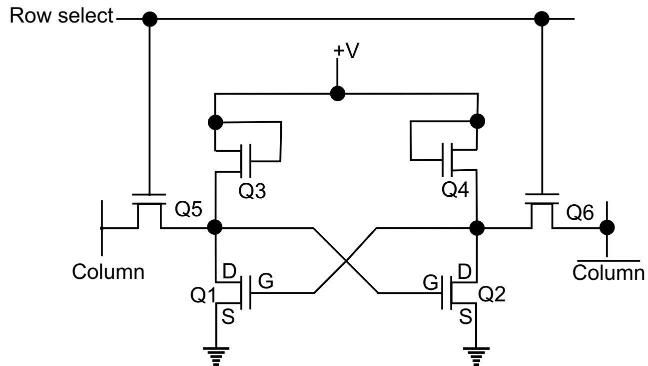


Fig. 8.6: Basic six-transistor static memory cell

Data is written into the cell by applying the data and its complement at the column and $\overline{\text{column}}$ inputs respectively, with Q_5 and Q_6 in the ON condition. Data can be read out from the column line after Q_5 and Q_6 are enabled.

Some of the characteristics of a SRAM cell are:

- It is volatile—i.e., data is lost on switch-off.
- When powered, the cell may assume either 1 or 0 state.
- Easy interfacing possible.
- No special timing circuits required.
- Bit density is low compared to DRAM.

37. Draw a dynamic RAM (DRAM) cell and explain its operation.

Ans. A basic DRAM storage cell is shown below:

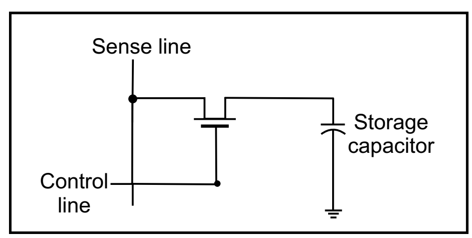


Fig. 8.7: Dynamic RAM

When column (sense) and row (control) lines go high, the MOSFET conducts and charges the capacitor. Again when the column and row lines go high, the MOSFET opens and the capacitor retains its charge. Thus, it can store a single bit. Since only a single MOSFET and a capacitor are employed to store a bit, the DRAM density is high. Here, the MOS transistor acts as a switch.

Some of the characteristics of a DRAM cell are

- Higher packing density.
- Charge leaks, thus refreshing needed.
- Extra hardware needed to implement refreshing operation.

38. Compare a SRAM and DRAM cell.

Ans. The comparison is shown in a table 8.4:

Table 8.4: Comparison between Static and Dynamic RAM cells

S.No.	Static memory		Dynamic memory
1.	Stored data is retained as long as power remains ON.	1.	Stored data gets lost and repeated refreshing is required.
2.	Stored data do not change with time.	2.	Stored data changes with time.
3.	Consumes more power.	3.	Consumes less power than static memory.
4.	Expensive.	4.	Less expensive.
5.	These memories have less packing density.	5.	Higher packing density.
6.	These memories are not easy to construct.	6.	Simpler in construction.
7.	No refreshing required and easy in operation.	7.	Refreshing required with additional memory circuitry and hence complicated operation.
8.	No maintenance.	8.	Maintenance needed.

39. What are the different types of DRAMS available. Briefly describe their operations.

Ans. The different types of DRAMS available are as follows: SDRAM, FPM DRAM, EDO DRAM, DDRS DRAM, SL DRAM, and DR DRAM.

Here is a brief description about the different DRAMS.

SDRAM: It stands for synchronous DRAM. Internally, these DRAMs are organised into two banks for very fast reading. Its internal circuit is quite complex because of various operations needed such as burst length, sequential or interleaved data, self-refreshing, $\overline{\text{CAS}}$ -before- $\overline{\text{RAS}}$, etc.

SDRAMs transfer data in rapid-fire bursts of several sequential memory locations on the same page. The first memory accessing requires maximum time because of latency problem. After that data are transferred and clocked out by the bus system clock.

FPM DRAM: It stands for fast page mode DRAM. Within the same page it allows fast random memory accessing. Fast memory accessing is possible since to access memory locations on the same page, only the lower address lines are to be changed.

EDO DRAM: It stands for extended data output DRAM. It is a bit faster than FPM DRAM. In the case of EDO DRAM, the memory controller can output the next address while the current word is being read out from its output lines. This is not so in case of FPM DRAM.

DDRS DRAM: It stands for double data rate SD RAM.

The doubling of data rate over SDRAM is because of transfer of data on the rising and falling edge of the system clock.

SL DRAM: It stands for synchronous link DRAM.

It is an improvement over DDRS DRAM. Like DDRS DRAM, it also transfers data on the rising and falling edge of system clock with bus speed of around 200 MHz.

DR DRAM: It stands for Direct Rambus DRAM.

It is a proprietory of Rambus Inc. It utilises a newer approach to DRAM technology and is currently under development. Within this DR DRAM, more control has been incorporated for relisation of newer technology.

40. Why DRAM chips do not have $\overline{CS}$ signal?

Ans. The $\overline{CS}$ (chip select signal, normally low) signal either selects or deselects a chip as per its status (0 selects and 1 deselects). DRAM chips have a $\overline{RAS}$ (row address strobe) and a $\overline{CAS}$ (column address strobe), which together perform the function of $\overline{CS}$ signal.

Several address lines go to a row address decoder and several others go to a column address decoder. The output of these two decoders ($\overline{RAS}$ and $\overline{CAS}$) then select a particular cell of the corresponding row and column addresses in which read or write operation is then performed—like a static RAM.

41. What kinds of memory expansions are possible by combining memory chips?

Ans. By combining memory chips, two kinds of expansions are possible—one is word size expansion and the other is capacity expansion.

By word size expansion is meant that each memory location can be expanded say from 1 nibble to 1 byte, while by capacity expansion is meant the expansion of a 2 KB memory into, say a 8 KB or 16 KB memory unit.

42. What is the principle of operation of an optical memory?

Ans. An optical disk memory operates on the principle of reflection or scattering of a very narrow laser beam of the optical surface having microscopic size bubbles or pits. These represent logical 1's.

43. What are the advantages of optical memory?

Ans. These kind of memories have large memory, relatively low in cost, very less dust-prone and access time comparable to hard disks.

44. Name the basic types of optical disk memories.

Ans. The basic forms of optical disk memories are as follows:

⇒ WORM or Write once read memory
⇒ CD-ROM or Compact disk ROM
⇒ OROM or Optical ROM

Out of the above, OROM can be written several times, while the contents of CD-ROMs and WORMs cannot be erased.

45. Name several commercially available EPROMs.

Ans. Several EPROMs are available commercially, which are listed below in a tabular form. Out of these, the IC types 2708, 2716, 2732–although in use today, are increasingly being replaced by the newer varieties because of higher bit capacity.

ICs	No. of bits (in K)	Memory Organisation type	Erasing type	Output after erasing
2708	8	1 K × 8	UV erasable	FF H
2716	16	2 K× 8	”	”
2732	32	4 K × 8	”	”
2764	64	8 K × 8	”	”
27128	128	16 K × 8	”	”
27256	256	32 K × 8	”	”
27512	512	64 K × 8	”	”

46. Design a memory having size 16 × 8 from 16 × 4 memory modules.

Ans. A 16 × 8 memory module indicates that it can address 16 different memory addresses, each address location can store 1 byte of data/instruction. The design is given below:

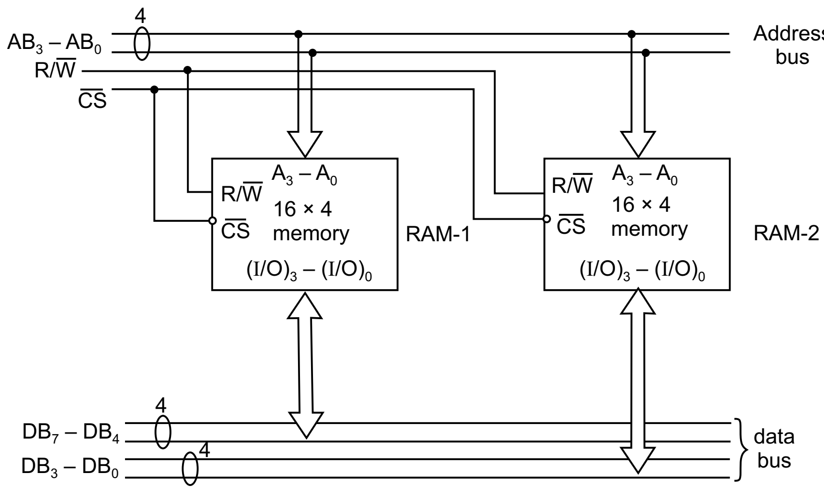


Fig. 8.8: Diagram showing a 16 × 8 memory obtained from two 16 × 4 memories

The address lines $AB_3 - AB_0$ can address 16 different addresses (from 0000 to 1111) and connected to the two RAMs as shown. Chip selection is on the basis of $\overline{CS}$ signal while $R/\overline{W}$ signal governs of whether reading (from memory) or writing (into the memory) is to be done.

Once a particular address has been selected (by $AB_3 - AB_0$ lines), $DB_7 - DB_4$ stores the upper nibble of data in the left RAM (RAM-1) while $DB_3 - DB_0$ stores the lower nibble of the data in the right RAM (RAM-2).

47. Develop a 32 × 4 memory module by combining two 16 × 4 memory chips.

Ans. The interconnections between the two 16 × 4 memory chips is shown below which yields a 32 × 4 memory module.

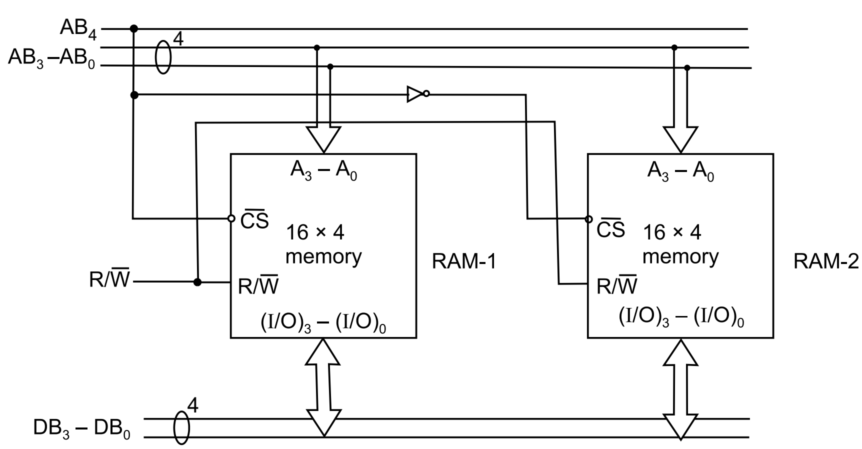


Fig. 8.9: Diagram showing a 32 × 4 memory obtained from two 16 × 4 memory modules

The address line AB_4 selects either RAM-1 or RAM-2. RAM-1 is selected when $AB_4 = 0$ while RAM-2 is selected when $AB_4 = 1$. The address ranges for RAM-1 is 00000 to 01111 while the same for RAM-2 is 10000 to 11111. Thus the combination acts as a memory having 32 different addresses, each of which can store 1 nibble (4 bits) of data/instruction—controlled by $DB_3 - DB_0$ lines.

48. Explain 'memory foldback area'.

Ans. It can be explained with the help of µP 8085 which can address 64 KB. Out of the 16 address lines available, 3 address lines (say $AB_{15} - AB_{13}$) can be inputted to a 3 to 8 line decoder (it may be 74ALS138). Thus each of the 8 output decoded lines divides the entire 64 K into 8 nos. of 8 K address blocks. Suppose in the 8 K address block 2000–3FFF, a single 2 K × 8 RAM is used intended for operation from 2000–27FF. Since this is a 2 KB of memory, it can be addressed by 11 address lines (say $AB_{10} - AB_0$). Now since the decoder output is active for 8 K addresses, hence the RAM will also respond to the remaining 6K addresses—resulting in the same contents of RAM to appear at the three address blocks 2800–2FFF, 3000–37FF and 3800–3FFF. Thus, the areas of the memory (here 2800 to 3FFF —a 6 K block) which are redundantly occupied by a device because of incomplete address decoding is known as 'memory foldback area'.

49. Mention the facilities available when power fails in a µC based system.

Ans. A 'power-fail' situation may occur at any time of a micro computer's operations. Critical applications—such as in process control systems, intelligent terminals, printers demand a stand by power supply when the main power fails.

There are several methods by virtue of which critical data can be saved in power fail situations. These are:

1. Back up batteries can hold critical data in RAMs when power fails, but problem arises if power remains off for a considerable time, i.e., the back up batteries are drained out after prolonged use and data will be lost.
2. Sometimes critical data are stored in non-volatile flash memories—thus no battery back up is needed. Thus in this case no problem is there even if power remains off for a considerable time. But the associated problem of a flash chip remains—i.e., erasing or writing one or few bytes is not possible, rather it is done sectorwise at a time.
3. The third approach involves storing of all critical data in high speed RAMs. When power fails, the CPU executes a short power down program stored in ROM. This allows the critical data from high speed RAMs to CMOS RAMs which are provided with battery back up. The 'power down program' is invoked by a circuit that senses the power failure situation.

On restoration of power, the CPU executes a power up program so that all critical data stored in back up devices are transferred back to system RAM—This is true for the above mentioned three cases.

50. Draw a typical memory read cycle and explain.

Ans. A typical memory read cycle is shown in Fig. 8.10.

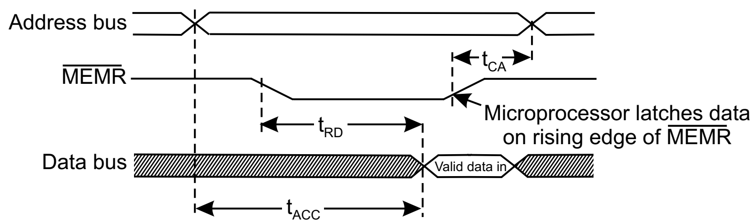


Fig. 8.10: Typical memory read cycle for a microprocessor. The memory must respond with valid data t_{ACC} seconds after the memory address is placed on the bus by the processor

The memory read cycle begins by outputting a valid address on its $A_0 - A_{15}$ address lines, after which the $\overline{MEMR}$ (active low) signal goes low, indicating that it is a memory read cycle. Then the microprocessor turns its internal data bus around so that it is now ready to receive data from the designated memory address.

The memory then places valid data on the data bus before the rising edge of $\overline{MEMR}$. On the rising edge of the $\overline{MEMR}$ signal, the data is latched into the microprocessor signalling the end of the memory read cycle.

The three time periods shown in the figure are explained here.

t_{ACC} : This is the address access time. It corresponds to the maximum amount of time the memory needs to decode the address and place the selected data byte on the data bus.

t_{RD} : This is the maximum amount of time after $\overline{MEMR}$ goes low that valid data is placed on the data bus by the designated memory.

t_{CA} : This is the minimum amount of time after $\overline{MEMR}$ goes high before a new address appears on the address bus. If this minimum time is not met, it may be possible that a new memory address have started to put a new data byte on the data bus while the previous memory address was still outputting its own data. This will result in 'bus contention'.

51. Draw a typical memory write cycle and explain.

Ans. A typical memory write cycle is shown below:

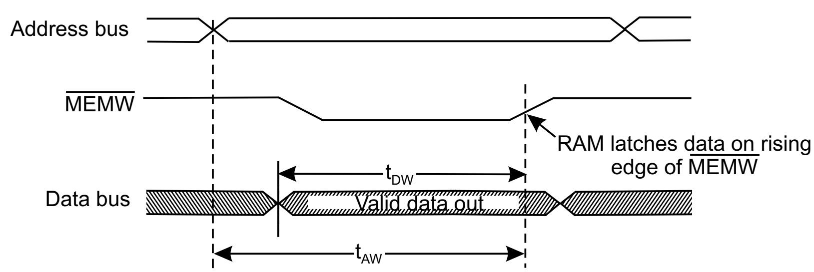


Fig. 8.11: Typical memory write cycle for a microprocessor. The memory must latch the data word within t_{AW} seconds after its address is output by the microprocessor

The memory write cycle begins by outputting a valid address on its $A_0 - A_{15}$ address lines. Valid data is outputted on $D_7 - D_0$ lines early in the machine cycle which

corresponds to 'data write set up time'. After this, the $\overline{MEMW}$ line goes low indicating the memory write cycle.

The memory now has time up to the rising edge of $\overline{MEMW}$ to latch the data.

The two time periods shown in the figure are explained here.

t_{AW} : This is the minimum amount of time that valid address will be held on the bus before $\overline{MEM}$ goes high.

t_{DW} : This is the minimum amount of time that valid data will be held on the bus before $\overline{MEM}$ goes high.

52. Show how the $\overline{MEMR}$ and $\overline{MEMW}$ signals are derived from IO/$\overline{M}$, $\overline{RD}$ and $\overline{WR}$ signals of µP 8085.

Ans. The figure below shows the generation of $\overline{MEMR}$ and $\overline{MEMW}$ signals, along with $\overline{IOR}$ and $\overline{IOW}$ signals.

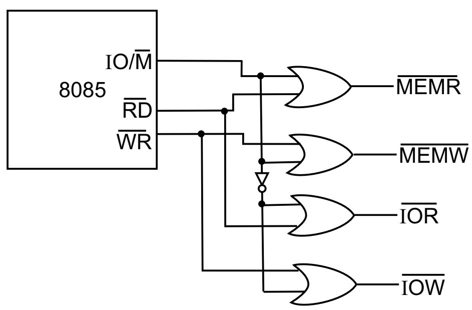


Fig. 8.12: Generation of $\overline{MEMR}$, $\overline{MEMW}$ along with $\overline{IOR}$, $\overline{IOW}$ Signals

53. Explain the working of a FIFO (first in first out) memory system.

Ans. In a RAM FIFO, the data word which is entered first is the one which is read out first. The FIFO memory operation (reading or writing) is controlled by a address pointer register that keeps a track of location where from data are to be written and also the location where from data are to be read.

A FIFO RAM is used as a data rate buffer and is really useful for situations where data takes place at widely diverging rates—e.g., from a computer to a printer or a keyboard to the computer. In case of printing, data meant for printing are first stored in a FIFO memory inside the printer. The printer then reads out the FIFO memory at its own rate and printing done in the same order in which they were sent by the computer.

Data rate buffers are also known as linear buffers.

54. How do circular buffers differ from linear buffers?

Ans. A circular buffer is a memory system which stores the 'last' n values entered, where n is the number of memory locations in the buffer.

A circular buffer, whose memory capacity is n, is addressed by a mod-n address counter. Thus, when the mod-n counter reaches the last (or highest) memory location, the next one to be addressed by the same would be the first address location.

Digital signal processing and digital filtering employ circular buffers as memory because their calculations are based on the recently taken data values.

55. What is usually meant by the speed of a memory?

Ans. Usually, the term 'speed of a memory' is meant the time needed to access the memory. This is also known as 'access time'.

56. Why it is necessary to decode an address from the microprocessor?

Ans. The address (to either memory or I/O device) is given out by the microprocessor. It is to be remembered that microprocessor is a sequential device—meaning that it can communicate (read or write) with only one device at any given instant. This is because the data bus, address bus and the control bus for all the devices connected to the microprocessor are common. Thus to communicate properly with a device (either memory or I/O), decoding of address is a must. Thus address decoding pinpoints a particular memory location or I/O.

57. What are the common address decoding techniques?

Ans. There are two kinds of address decoding techniques. These are

- Absolute Decoding—also known as Full Decoding.
- Linear Decoding—also known as Partial Decoding.

58. Describe the absolute decoding scheme.

Ans. In this scheme, all the higher order address lines are decoded to select the memory chips and a particular memory chip is selected only for a specified logic level on these address lines. The particular memory chip is not selected for any other logic level. The three higher order address lines go to the decoder 74LS138 which outputs Y_0 .... Y_7. While a low on Y_0 selects only RAM-1, a low on Y_7 selects RAM-8. Thus depending on the address line status on $A_{15} - A_{13}$, only one memory chip is selected.

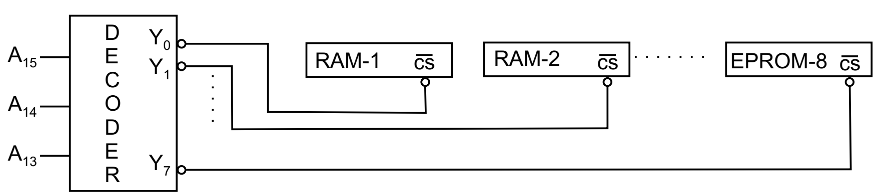


Fig. 8.13: Absolute decoding scheme

59. Describe the linear decoding scheme.

Ans. In this scheme, the decoder logic is eliminated by using individual high order address lines to select particular memory chip. As shown in the figure, when $A_{15} = 1$, ROM-2 is selected while RAM-1 is selected for $A_{15} = 0$.

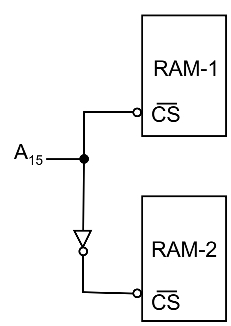


Fig. 8.14: Linear decoding scheme

60. Compare absolute and linear decoding schemes.

Ans. The comparison between the two schemes are tabulated below in Table 8.5:

Table 8.5: Comparison between Full and Partial Address Decoding

S. No.	Full address decoding (Absolute decoding)		Partial address decoding (Linear decoding)
1.	Used in large memory systems.	1.	Used in small memory systems.
2.	All higher order address lines are decoded to select the memory or I/O device.	2.	A few high order address lines are utilised to select individual memory or I/O chips.
3.	Decoding logic requires more hardware.	3.	Hardware requirement for the same is very small and sometimes not required at all in very small dedicated systems.
4.	No multiple addresses.	4.	Suffers from the drawback of multiple addresses (also called shadow addresses).
5.	Higher cost of decoding.	5.	Lesser cost of decoding.
6.	Future memory expansion is easier.	6.	Future memory expansion is difficult.
7.	No bus contention problem.	7.	May suffer from bus contention problem—may occur if more than one memory chip gets selected because of wrong address generation.

61. What are meant by system memories and standard memories?

Ans. Some CPU manufacturers design memory chips compatible with their CPUs. This is done to reduce burden on extra hardware. Such memories are called system memories.

Again so many memory chips are manufactured which can be used with different varieties of CPUs after overcoming the incompatibility problems. Such memories are called standard memories.

62. Compare system memories with standard memories.

Ans. The comparison between system memories and standard memories is tabulated below:

Table 8.6: Comparison between System Memory and Standard Memory

S. No.	System memories		Standard memories
1.	More expensive.	1.	Less expensive.
2.	May not be available from multiple vendors.	2.	May be available from multiple vendors.
3.	Not available in a variety of organisations, sizes and speeds.	3.	Available in a variety of organisations, sizes and speeds.
4.	With a particular CPU, full compatibility is there with respect to bus and speed. Thus interfacing is easier and less hardware needed.	4.	For compatibility with respect to bus and speed, extra hardware is needed. Thus interfacing is relatively complex.
5.	Easy to choose a system memory because of standardisation.	5.	Difficulty in choosing a particular memory from a variety of memory chips available from many manufacturers.

63. What is the advantage of memory interlacing technique?

Ans. It increases the effective speed of memory.

64. What advantage is derived when larger RAM is inserted in a μC based system?

Ans. It enhances speed by eliminating the need for external memory.

9. PERIPHERAL CHIPS

9a

8255: Programmable Peripheral Interface

1. Draw the pin diagram of PPI 8255.

Ans. The pin diagram of 8255 is shown in Fig. 9a.1

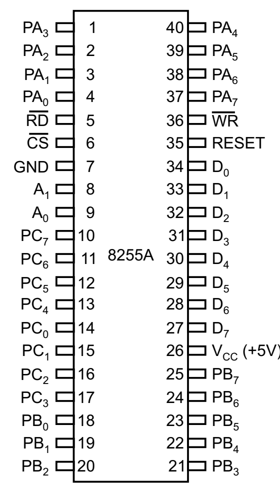


Fig. 9a.1: 8255 Pin diagram (*Source:* Intel Corporation)

2. Draw the block diagram of 8255.

Ans. The block diagram is shown in Fig. 9a.2.

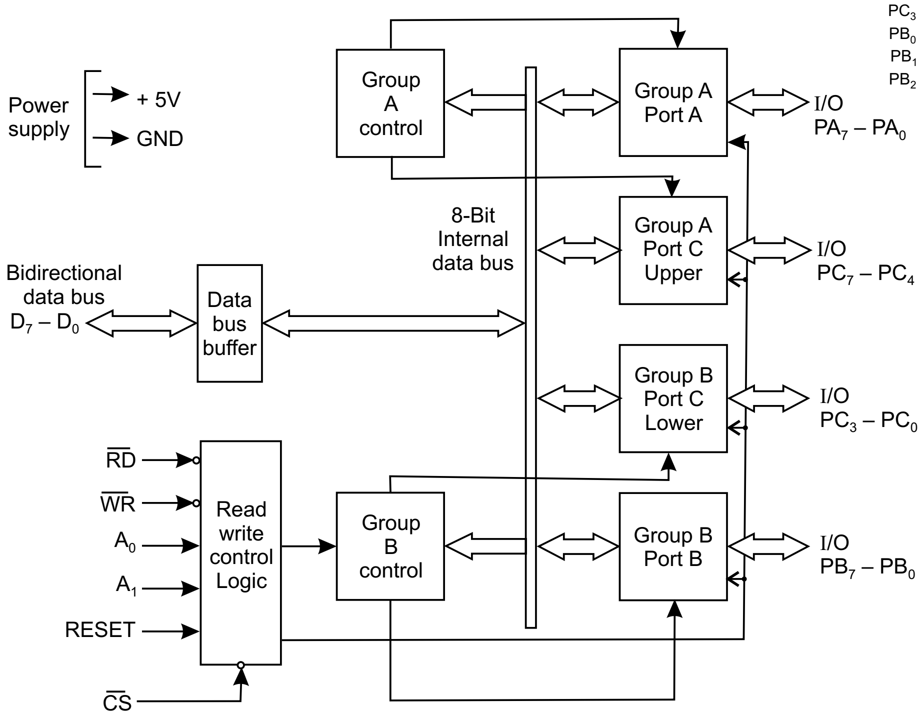


Fig. 9a.2: Block diagram of 8255 (*Source:* Intel Corporation)

3. How many ports are there in 8255 and what are they?

Ans. Basically there are three ports in 8255, viz., Port A, Port B and Port C, each having 8 pins. Again Port C can be divided into Ports C_{upper} and Port C_{lower}—each having four pins i.e., a nibble. Thus 8255 can be viewed to have four ports—Port A, Port B, Port C_{upper} and Port C_{lower}.

4. What pins are associated with Read/Write control logic block?

Ans. There are six pins associated with Read/Write control logic block. These are $\overline{CS}$, $\overline{WR}$, A_0, A_1, Reset and $\overline{CS}$ signals.

5. In how many modes can 8255 operate?

Ans. PPI 8255 can operate in three modes. (*a*) Mode 0 (*b*) Mode 1 and (*c*) Mode 2.

Apart from the above, there is another mode called BSR mode (Bit Set/Reset mode).

6. Distinguish between the three modes of 8255.

Ans. The three modes are Mode 0, Mode 1 and Mode 2. These are I/O operations and selected only if D_7 bit of the control word register is put as 1.

The three operating modes of 8255 are distinguished in the following manner:

Mode 0: This is a basic or simple input/output mode, whose features are:

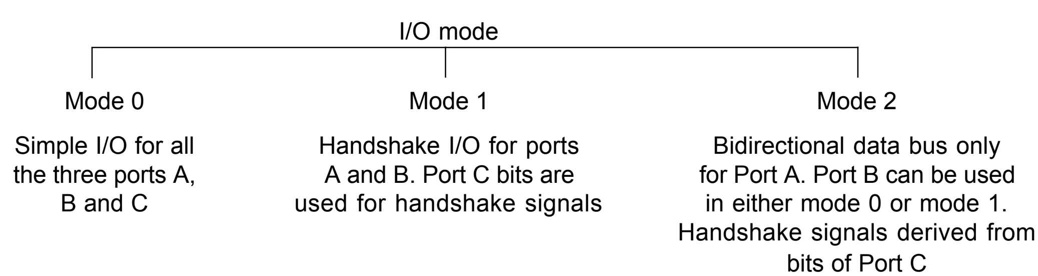
- Outputs are latched.
- Inputs are not latched.
- All ports (A, B, C_U, C_L) can be programmed in either input or output mode.
- Ports don't have handshake or interrupt capability.
- Sixteen possible input/output configurations are possible.

Mode 1: In this mode, input or outputting of data is carried out by taking the help of handshaking signals, also known as strobe signals. The basic features of this mode are:

- Ports A and B can function as 8-bit I/O ports, taking the help of pins of Port C.
- I/Ps and I/Ps are latched.
- Interrupt logic is supported.
- Handshake signals are exchanged between CPU and peripheral prior to data transfer.
- In this mode, Port C is called status port.
- There are two groups in this mode—group A and group B. They can be configured separately. Each group consists of an 8-bit port and a 4-bit port. This 4-bit port is used for handshaking in each group.

Mode 2: In this mode, Port A can be set up for bidirectional data transfer using handshake signals from Port C. Port B can be set up either in mode 0 or mode 1.

The basic operations of the three modes are shown below:



7. Which word determines the operating mode of 8255?

Ans. A single control word determines the operating mode of 8255.

8. For data transfer using 8255, when mode 0 should be selected?

Ans. When unconditional or non-handshaking I/O is required, mode 0 is chosen.

9. How many categories of handshake signals are there? Which is advantageous?

Ans. Handshake signals can be used with either (1) status check I/O or (2) Interrupt I/O.

In the status check I/O, the CPU gets tied up in a loop until the status of the I/O becomes ready while it is the I/O device which interrupts the CPU in interrupt I/O.

10. Explain how the different ports and control words are selected for 8255.

Ans. The two address lines, along with $\overline{CS}$ signal, determine the selection of a particular port or control register. This is explained below:

$\overline{CS}$	A_1	A_0	*Selection*
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control Register
1	X	X	8255 not selected (because $\overline{CS}$ = 1)

$\overline{CS}$ signal is made 0 by choosing $A_7 = 1$ and A_6 though $A_2 = 0$

Thus,

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0			
1	0	0	0	0	0	0	0	=	80_H	Port A selected
1	0	0	0	0	0	0	1	=	81_H	Port B selected
1	0	0	0	0	0	1	0	=	82_H	Port C selected
1	0	0	0	0	0	1	1	=	83_H	Control Register selected

11. What is BSR mode and what are its characteristics?

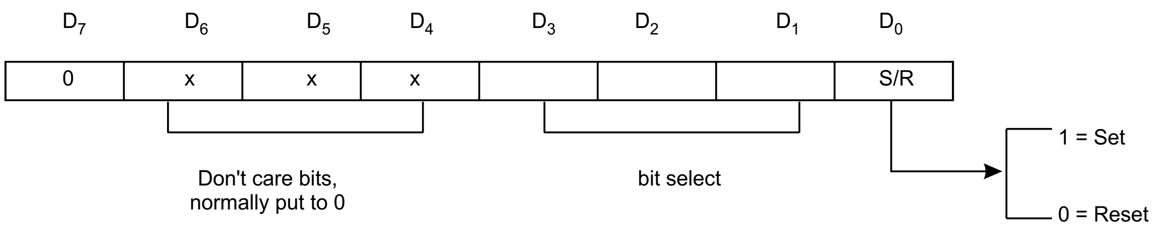
Ans. BSR mode stands for Bit Set Reset mode.

The characteristics of BSR mode are:

- BSR mode is selected only when $D_7 = 0$ of the Control Word Register (CWR).
- Concerned with bits of port C.
- Individual bits of Port C can either be Set or Reset.
- At a time, only a single bit of port C can be Set or Reset.
- Is used for control or on/off switch.
- BSR control word doesn't affect ports A and B functioning.

12. Discuss the control word format in the BSR mode.

Ans. The content of the control word register will be as follows, when used in the BSR mode and selects (either Sets or Resets) a particular bit of Port C at a time.



D_3	D_2	D_1	*Particular bit of Port C selected*
0	0	0	Bit 0
0	0	1	Bit 1
0	1	0	Bit 2
0	1	1	Bit 3
1	0	0	Bit 4
1	0	1	Bit 5
1	1	0	Bit 6
1	1	1	Bit 7

Fig. 9a.3: The CWR in the BSR mode

13. Write a BSR control word to set bits PC_7 and PC_0 and to reset them after 1 second delay.

Ans. To set or reset any particular bit of Port C in the BSR mode, the control word register is to be appropriately loaded. The above is done by loading the accumulator and sending the same to the control register (i.e., by sending the same to the address of the control register). The address of control word register (CWR) is 83_H.

Program:

MVIA, $0F_H$	(Accumulator loaded with $0F_H$ to set PC_7 bit of Port C)
OUT 83_H	(This sets PC_7 bit of Port C)
MVIA, 01H	(Accumulator loaded with 01_H to set PC_0 bit of Port C)
OUT 83_H	(This sets PC_0 bit of Port C)
CALL DELAY	(Assume the DELAY is for 1 second)
MVIA, 00H	(Accumulator loaded with 00_H to reset PC_0 bit of Port C)
OUT 83_H	(This resets PC_0 bit of Port C)
MVIA, $0E_H$	(Accumulator loaded with $0E_H$ to reset PC_7 bit of Port C)
OUT 83_H	(This resets PC_7 bit of Port C)

14. Show the control word format for I/O mode operation of PPI 8255.

Ans. The control word format, when 8255 is operated in I/O mode, is shown below:

For 8255 PPI to be operated in I/O mode, D_7 bit must be 1.

The three ports are clubbed into two groups—Groups A and B. Group A consists of Port A and C_U. Port A can be operated in any of the modes—0, 1 or 2. Group B consists of Port B and C_L. Here Port B can be operated in either mode 0 or 1.

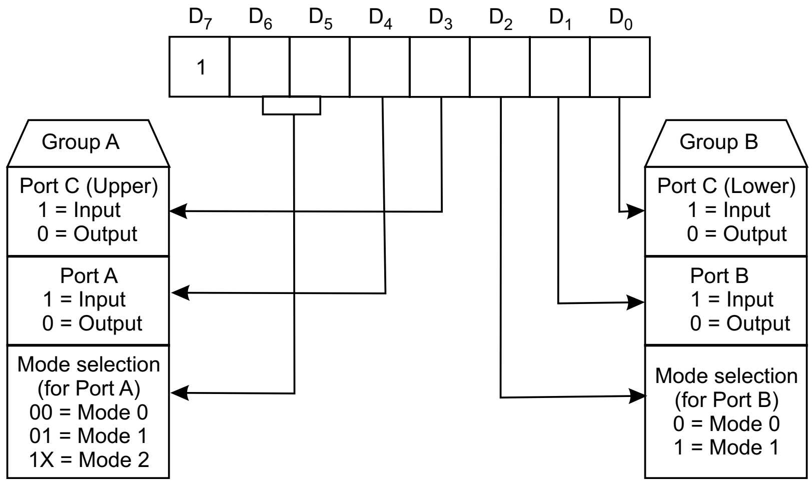


Fig. 9a.4: The CWR in the I/O mode

15. What happens when RESET pin of 8255 is made high?

Ans. When a 1 is applied on RESET pin of 8255, the three ports are put in the input mode. All flip-flops are cleared and interrupts are reset. This condition is not altered even when RESET goes low. 8255 can then be programmed in any mode by appropriately loading the control word register. The mode operation can be changed by altering the content of the control word register, whenever needed.

16. Write down the mode 0 control words for the following two cases:

(*a*) Port A = Input port, Port B = not used, Port C_U = Input port and Port C_L = Output port.

(*b*) Port A = Output port, Port B = Input port, Port C = Output port

Ans. The control words for the two cases will be as follows:

(*a*)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
1	0	0	1	1	0	0	0	=	98_H
I/O mode	Mode 0 for Port A		Port A input	Port C_U input	Port B not used		Port C_L output		

Thus, the control word would be = 98_H

(*b*)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
1	0	0	0	0	0	1	0	=	82_H
I/O mode	Mode 0 for Port A		Port A output	Port C_U output	Mode 0 for Port B	Port B input	Port C_L output		

Thus, the control word would be = 82_H.

17. In mode 1 what are the control signals when ports A and B act as input ports. Discuss the control signals. Draw the timing waveforms for such a strobed input.

Ans. The following are the control signals when ports A and B act as input ports (under mode 1) $\overline{\text{STB}}_A$, IBF_A, INTE_A for Port A and $\overline{\text{STB}}_B$, IBF_B, INTE_B for Port B, respectively. The details about the input control signals are discussed below:

- $\overline{STB}$ *(Strobe input):* This is an active low signal generated by a peripheral device. When a peripheral device has some valid data, it sends the same via Port A or B and sends a low $\overline{\text{STB}}$ signal. This data is accepted by 8255 and it generates a IBF and INTR (provided INTE is set previously).
- *IBF (Input buffer full):* On receipt of $\overline{\text{STB}}$ signal from peripheral device, data is stored in 8255 by its input latch. In its turn, 8255 generates a high IBF. IBF is reset when CPU reads the data.
- *INTR (Interrupt request):* This active high output signal is generated only if $\overline{\text{STB}}$, IBF and INTE are all set at the same time. This signal interrupts the CPU via its INTR (pin no. 10 of 8085).
- *INTE (Interrupt Enable):* This is an internal F/F which can be set/reset using the BSR mode. It must be set if INTR signal is to be effective.

The following figure shows Port A in Mode 1 (input), along with the timing diagrams.

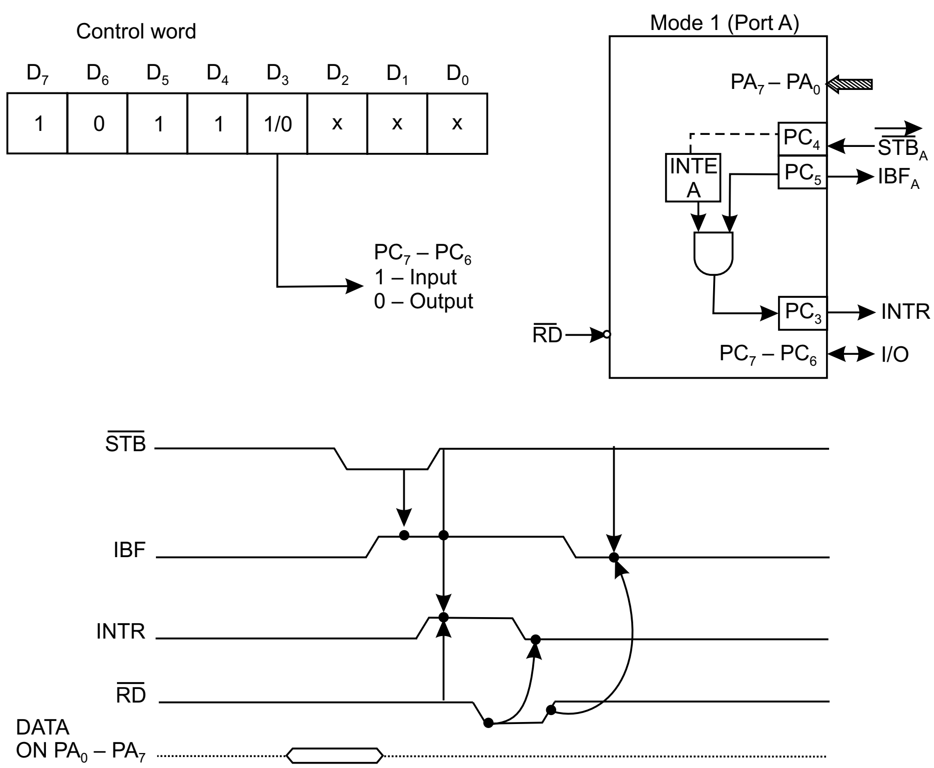


Fig. 9a.5: Port A in Mode 1 (Input) (*Source:* Intel Corporation)

18. In mode 1, what are the control signals when ports A and B act as output ports. Discuss the control signals. Draw the timing waveforms for such a strobed output.

Ans. The following are the control signals when ports A and B act as output ports (under mode 1) $\overline{OBF_A}$, $\overline{AC\ \ }_A$, $INTE_A$ for Port A and $\overline{OBF_B}$, $\overline{ACK_B}$, $INTE_B$ for Port B respectively.

The details about the output control signals are discussed below:

- $\overline{OBF}$ *(Output buffer full):* This is an active low output signal. This signal becomes low when the CPU writes data into the output latch of 8255. This output signal from 8255, which goes to a peripheral, indicates to the peripheral that the data on the output latch of 8255 is ready to be read.
- $\overline{ACK}$ *(Acknowledge):* When data reading by the peripheral from the output latch of 8255 is complete (i.e., the peripheral has accepted the data), it (the peripheral) outputs a low signal which is connected to the $\overline{ACK}$ (input signal) pin of 8255. On receipt of this low signal by 8255 (from peripheral), the $\overline{OBF}$ line of 8255 goes high.
- *INTR (Interrupt):* This signal is set only if $\overline{OBF}$, $\overline{ACK}$ and INTE (internal F/F) are all at high(1) state. This output signal from 8255 goes to INTR (pin 10 of 8085) to interrupt the CPU. The INTR signal is reset on the falling edge of $\overline{WR}$.
- *INTE (Interrupt Enable):* This is an internal F/F which can be set/reset in BSR mode. This must be set if INTR signal is to be effective.

The following figure shows Port A in mode 1 (output), along with the timing waveforms.

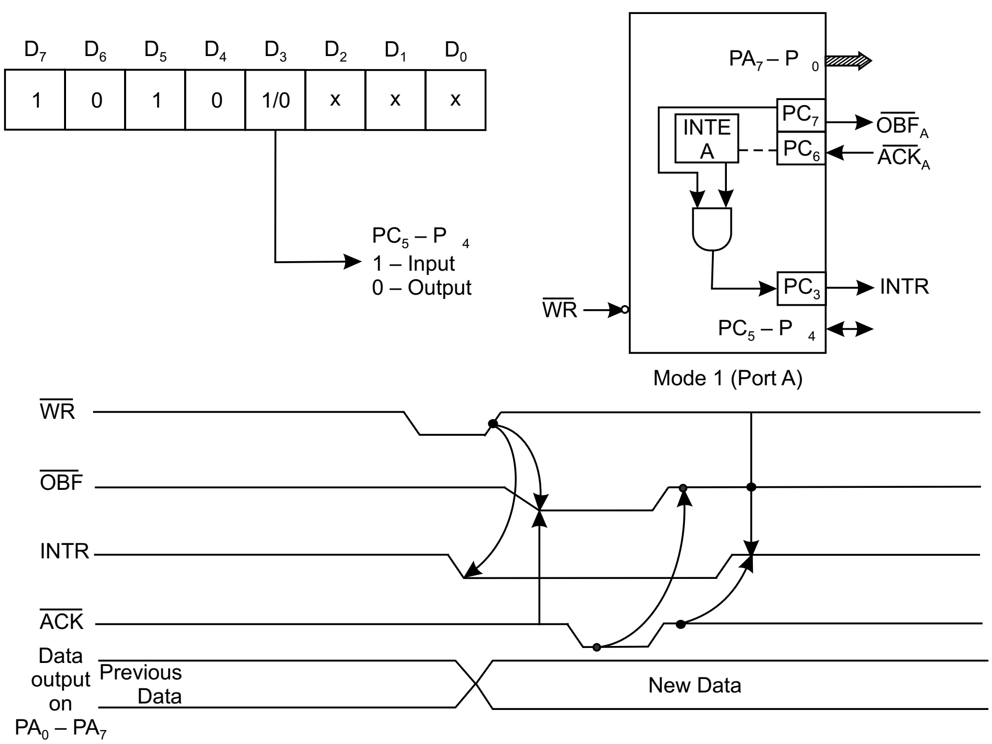


Fig. 9a.6: Port A in mode 1 (Output) (*Source:* Intel Corporation)

19. In mode 1, what are the methods available for data transfer? Which method is advantageous?

Ans. In mode 1, data transfer is possible involving 8255 when it is programmed to function either in (*a*) Status check I/O (also called Program Controlled I/O), (*b*) Interrupt I/O (also called Interrupt Controlled I/O).

The simplified flowcharts for the two schemes are shown below:

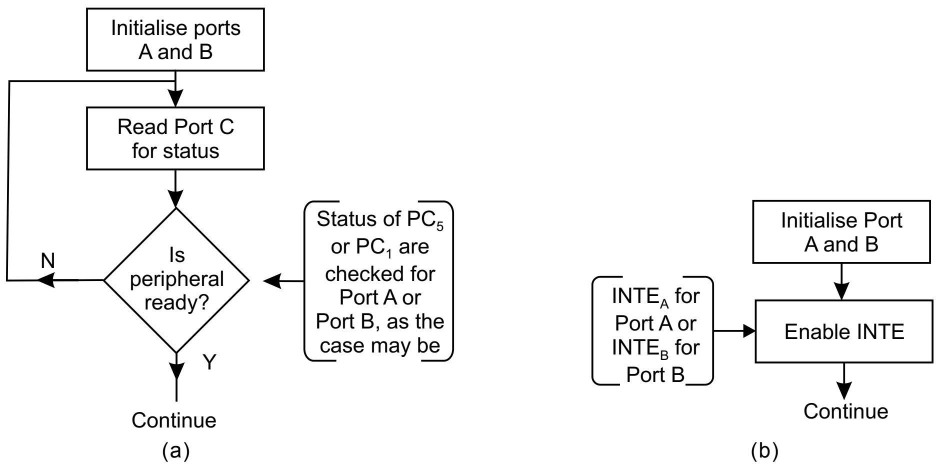


Fig. 9a.7: Flow charts for (a) Status check I/O, (b) Interrupt I/O

In status check I/O, CPU continues to check the status of IBF_A or IBF_B until they are high. This is done by reading the status word (port C) to check (PC_5 for IBF_A and PC_1 for IBF_B) for existence of IBF. It is known as 'polling' (reading) the status word. Here it is assumed that both Ports A and B act as input ports.

In interrupt I/O scheme, the status of either $INTR_A$ or $INTR_B$ (as the case may be) will have to be ascertained to know the port (A or B) which has requested (interrupted) for service. This is done by reading the status of $INTR_A$ (PC_3) or $INTR_B$ (PC_0) of the status word. Here again, it is assumed that both ports A and B act as input ports.

Status check I/O is disadvantageous because in this the CPU gets tied up in the loop until the IBF line (IBF_A or IBF_B, as the case may be) goes high.

20. Show the mode 1 status word format and discuss.

Ans. There are two mode 1 status word formats—one for input configuration and the other for output configuration. These are shown below:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O	I/O	IBF_A	$INTE_A$	$INTR_A$	$INTE_B$	IBF_B	$INTR_B$
Group A					Group B		

(a) Status Word for Mode 1 Input Configuration

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF}_A$	$INTE_A$	I/O	I/O	$INTR_A$	$INTE_B$	$\overline{OBF}_B$	$INTR_B$
Group A					Group B		

(b) Status Word for Mode 1 Output Configuration

Fig. 9a.8: Status word for mode 1 (a) Input (b) Output configuration

The mode 1 status format (either input or output configuration) can be read by an input read of Port C.

When 8255 is operated in mode 1, the processor has two choices — either polling or interrupt scheme—this is true irrespective of whether data is inputted to the CPU or otherwise.

When data is inputted (i.e., from peripheral to CPU), the CPU can poll (status check) the IBF line for presence of valid data. Else the processor can be interrupted via the INTR line. To know whether $INTR_A$ or $INTR_B$ has interrupted can be ascertained by reading the Port C for status of $INTR_A$ (bit D_3) or $INTR_B$ (bit D_0).

Again, when data is outputted (by CPU to 8255), again two choices are there. The CPU can poll the $\overline{OBF}$ or else be interrupted by INTR line.

If interrupt driven scheme is followed, then INTE (either $INTE_A$ or $INTE_B$) has to be previously set in BSR mode.

A confusion arises when interrupt driven I/O scheme is undertaken. For input configuration, it is seen that $\overline{STB_A}$ signal is connected to PC_4 and $INTE_A$ is controlled by PC_4. For Port B, the corresponding bit is PC_2—i.e., $\overline{STB_B}$ is connected to PC_2 and $INTE_B$ is also controlled by PC_2. But this poses no problem because INTE is set/reset in BSR mode and the BSR control word has no effect when ports A or B are set in mode 1.

21. Discuss the mode 2 of PPI 8255 in brief.

Ans. This mode is usually used for transferring data between two computers.

When operated in mode 2, only Port A can be used as a bidirectional 8-bit I/O bus, using $PC_3 – PC_7$ for handshaking. Port B can be programmed only in mode 0 ($PC_0 – PC_2$ as input or output) or in mode 1 ($PC_0 – PC_2$ used as handshaking signals).

22. How many possible combinations of 8255 would be there when it is operated in mode 2.

Ans. Only Port A can be operated in mode 2 and Port B in either mode 0 or mode 1.

Thus, there would be four possible combinations in mode 2. These are:

(*a*) Mode 2 and Mode 0 (input)
(*b*) Mode 2 and Mode 0 (output)
(*c*) Mode 2 and Mode 1 (input)
(*d*) Mode 2 and Mode 1 (output)

23. Discuss the Mode 2 control word.

Ans. The mode 2 control word is as shown below:

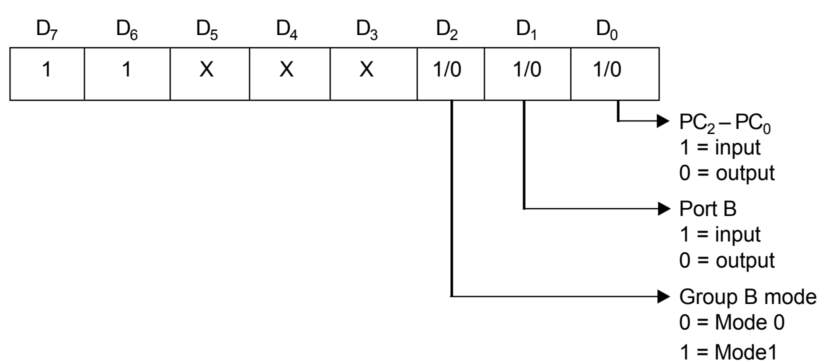


Fig. 9a.9: CWR in Mode 2

This control word is to be loaded into the control port to configure 8255 in mode 2. Bit D_0 of control port determines the I/O operations of $PC_2 - PC_0$. D_1 bit indicates the Port B input/output operation whereas bit D_2 determines Group B to be either in mode 0 or in mode 1 operation.

24. Draw Port A and the associated control signals when 8255 is operated in mode 2.

Ans. This is shown in the following figure:

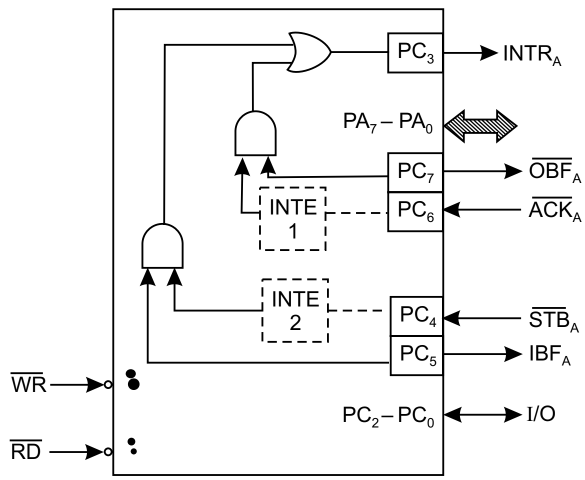


Fig. 9a.10: Mode 2 operation (*Source:* Intel Corporation)

25. What are the output control signals in mode 2 and discuss them?

Ans. The output control signals in mode 2 are $\overline{OBF}$, $\overline{ACK}$ and INTE 1. These control signals are discussed below: $\overline{OBF}$ stands for output buffer full, an active low signal. In its active condition (i.e., low), it indicates that the CPU has written data into Port A.

$\overline{ACK}$, an active low signal, stands for acknowledge. This acknowledgement signal is generated by a peripheral and it enables the tri-state output buffer or Port A and makes Port A data available to the peripheral.

INTE 1 is an internal F/F associated with output buffer full. INTE 1 can be used to enable or disable the interrupt (INTR) by setting or resetting PC_6 in BSR mode.

26. What are the input control signals in mode 2 and discuss them?

Ans. The input control signals in mode 2 are $\overline{STB}$, IBF and INTE 2. These control signals are discussed below:

$\overline{STB}$ (strobe input), an active low signal, enables Port A to latch the data available at its input. This happens if $\overline{STB} = 0$.

IBF (Input buffer full), an active high signal, indicates the data has been loaded into the input latch of Port A. This happens if IBF = 1.

INTE 2 is an internal F/F associated with input buffer full. INTE 2 can be used to enable or disable the interrupt (INTR) by setting or resetting PC_4 in BSR mode.

27. Draw a table that summarises the pin functions of Ports A, B and C for various modes of operation.

Ans. The following table shows the pin summary of Ports A, B and C for various modes of operation.

Table 9a.1: Pin summary for all modes of operation (*Source*: Intel Corporation)

	Mode 0		Mode 1		Mode 2
	In	**Out**	**In**	**Out**	**Group A only**
PA_0	In	Out	In	Out	↔
PA_1	In	Out	In	Out	↔
PA_2	In	Out	In	Out	↔
PA_3	In	Out	In	Out	↔
PA_4	In	Out	In	Out	↔
PA_5	In	Out	In	Out	↔
PA_6	In	Out	In	Out	↔
PA_7	In	Out	In	Out	↔
PB_0	In	Out	In	Out	—
PB_1	In	Out	In	Out	—
PB_2	In	Out	In	Out	—
PB_3	In	Out	In	Out	—
PB_4	In	Out	In	Out	—
PB_5	In	Out	In	Out	—
PB_6	In	Out	In	Out	—
PB_7	In	Out	In	Out	—
PC_0	In	Out	$INTR_B$	$INTR_B$	I/O
PC_1	In	Out	IBF_B	$\overline{OFB}_B$	I/O
PC_2	In	Out	$\overline{STB}_B$	$\overline{ACK}_B$	I/O
PC_3	In	Out	$INTR_A$	$INTR_A$	$INTR_A$
PC_4	In	Out	$\overline{STB}_A$	I/O	$\overline{STB}_A$
PC_5	In	Out	IBF_A	I/O	IBF_A
PC_6	In	Out	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$
PC_7	In	Out	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$

(Port B, PB_0–PB_7: Mode 0 or Mode 1 only)

28. What are the status of Port A outputs until they are enabled.

Ans. Port A outputs remain in the tri-state condition until they are enabled. This enabling is done by a low on the $\overline{ACK}$ signal generated by a peripheral.

29. On which line interrupts are generated in mode 2 for both input and output operations?

Ans. The same ($INTR_A$) is generated on PC_3 line, for both input and output operations.

30. Draw the status word in mode 2 and discuss.

Ans. The status word in mode 2 is as shown:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF_A}$	INTE1	IBF_A	INTE2	$INTR_A$	X	X	X
Group A					Group B		

Fig. 9a.11: Status word in mode 2

This status word is accessed by reading Port C. $D_7 - D_3$ bits of the status word carry the status of $\overline{OBF_A}$, INTE 1, IBF_A, INTE 2, $INTR_A$. The status of the remaining three bits i.e., $D_2 - D_0$ depend on the mode setting of Group B. If Group B is programmed to be in mode 0, then $D_2 - D_0$ are simply $PC_2 - PC_0$ (simple I/O). But if Group B is in mode 1, then the three bits $D_2 - D_0$ carry the information about the control signals for Port B—and they depend on whether B is acting as an input port or output port.

31. What are the types of devices with which data transfer takes place via the 8255 PPI?

Ans. The input devices from which data are read and delivered to the microprocessor via the input ports of 8255 PPI are ADCs, keyboards, control signals from process devices, paper tape readers, etc.

The output devices to which data are delivered via the output ports of 8255 are DACs, printers, video display, plotters, etc.

32. What kind of functions do the input and output ports play?

Ans. A port, whether an input or an output port, is a set of D F/Fs consisting of several pins in parallel. When used as input pins, they act as buffers and when used as output pins, they act as latches.

9b

8155/8156: Programmable I/O Ports and Timer

1. In what way 8155 and 8156 differs?

Ans. The Chip Enable (CE) signal is active low for 8155, whereas it is active high for 8156.

2. What are the essential features of 8155.

Ans. The essential features of 8155 are

- 8-bit 256 word RAM memory
- Two programmable 8-bit I/O port
- One programmable 6-bit IO port
- One programmable 14-bit binary Timer/Counter
- An internal address latch
- A control/status (C/S) register
- An internal decoder.

3. Functionally, how many sections are there in 8155?

Ans. Functionally, it has two sections—(*a*) a R/W memory and (*b*) programmable I/O and timer section.

4. Is it necessary to demultiplex the lower order bus AD_7 –AD_0 externally for 8155 to be connected to 8055?

Ans. No, it is not. This is because ALE, IO/$\overline{M}$, $\overline{RD}$ and $\overline{WR}$ signals of 8085 can be connected directly with 8155.

5. Draw the pin diagram of 8155.

Ans. The pin diagram of 8155 is shown in Fig. 9b.1.

6. Draw the functional block diagram of 8155.

Ans. The functional block diagram of 8155 is shown in Fig. 9b.2 :

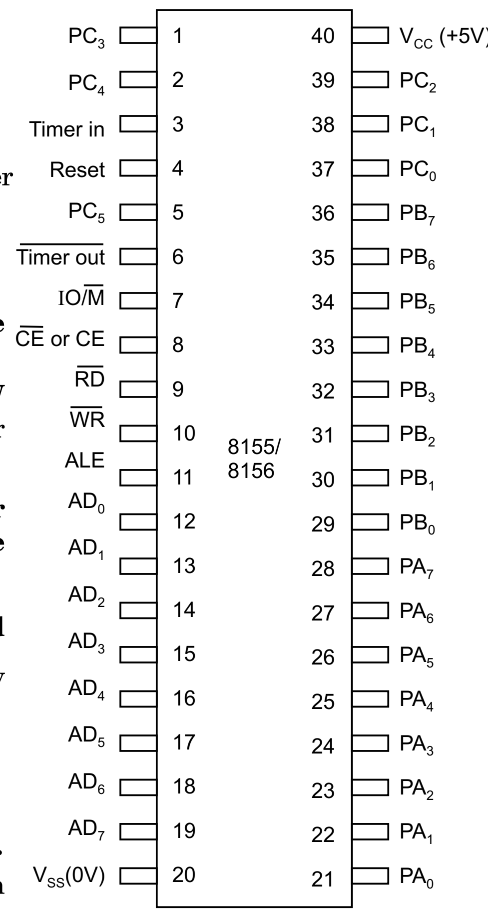


Fig. 9b.1: Pin diagram of 8155 (*Source:* Intel Corporation)

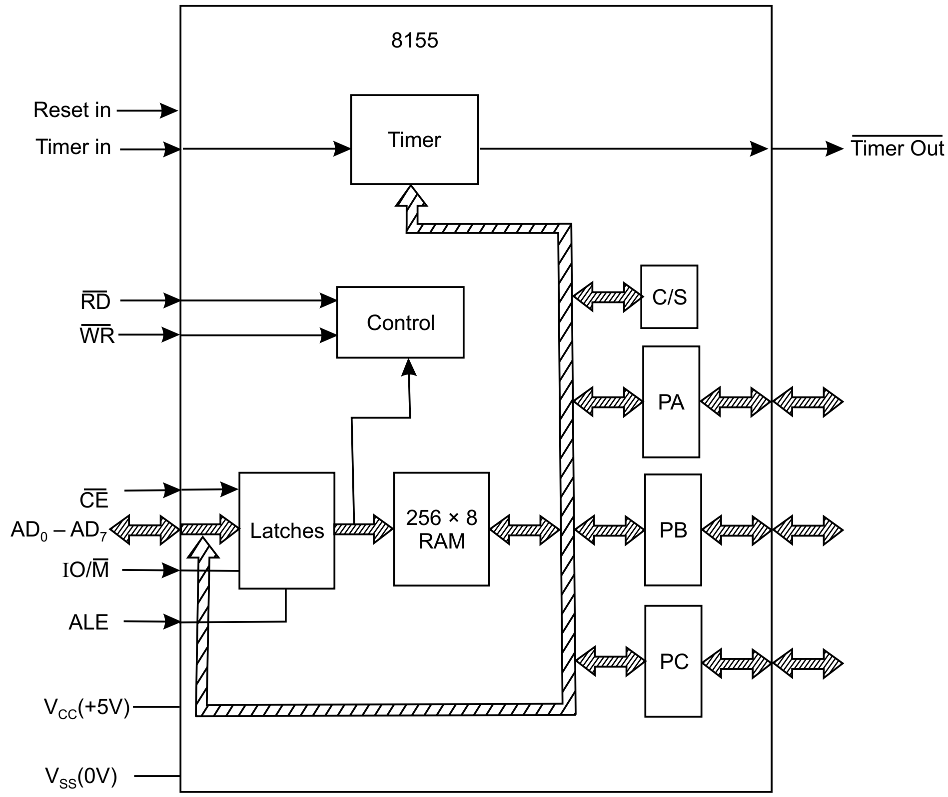


Fig. 9b.2: Functional diagram of 8155 (*Source:* Intel Corporation)

7. How the different ports, control/status register, timers are accessed? Write their addresses also.

Ans. The different combinations on the address lines A_2, A_1, A_0 select one of the above, as shown:

A_2	A_1	A_0		
0	0	0	⇒	Control/Status Register
0	0	1	⇒	Port A
0	1	0	⇒	Port B
0	1	1	⇒	Port C
1	0	0	⇒	LSB Timer
1	0	1	⇒	MSB Timer

The other five (viz., A_7 to A_3) on the address lines are as: 0 0 1 0 0. Thus

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		Address		Register/Port/Timer
0	0	1	0	0	0	0	0	⇒	20_H	—	Control/Status register
0	0	1	0	0	0	0	1	⇒	21_H	—	Port A
0	0	1	0	0	0	1	0	⇒	22_H	—	Port B
0	0	1	0	0	0	1	1	⇒	23_H	—	Port C
0	0	1	0	0	1	0	0	⇒	24_H	—	LSB timer
0	0	1	0	0	1	0	1	⇒	25_H	—	MSB timer

It is to be noted that the control/status register is having the same address 20_H, but the control register is accessed with $\overline{WR}$ = 0 and $\overline{RD}$ = 1. For status register access, $\overline{WR}$ = 1 and $\overline{RD}$ = 0. The control register can never be read. For any future reference, the control register content is stored in some accessible memory location.

8. Draw the control word format and discuss the same in detail.

Ans. The control word loaded in the control register configures the different ports and the timer of 8155. The control word format is shown below:

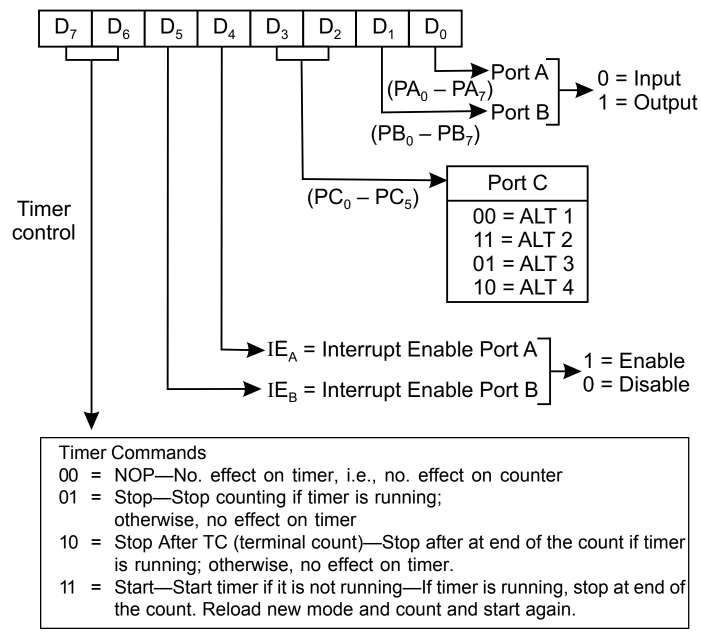


Fig. 9b.3: The control word format

The control register contains eight latches. The content of the lower 2 bits, viz., $D_1 – D_0$ configure ports A and B as input/output. Bits D_3 and D_2 configure bits $PC_0 – PC_5$ of port C (Port C is a 6-bit port while ports A and B both are of 8-bits) and can have four combinations—ALT1, ALT2, ALT3, ALT4 depending on the combinations of D_3 and D_2. Bits D_5 and D_4 are enable/disable pins for ports A and B respectively which enable/disable the internal flip-flop of 8155. Bits D_7 and D_6 contain the timer commands.

As already mentioned, combinations of $D_3 – D_2$ bits give rise to ALT1 to ALT4 modes, which assigns port C bits in different configurations and shown below:

Table 9b.1: Port C pin assignment (*Source:* Intel Corporation)

Pin	ALT1	ALT2	ALT3	ALT4
PC0	Input Port	Output Port	A INTR (Port A Interrupt)	A INTR (Port A Interrupt)
PC1	Input Port	Output Port	A BF (Port A Buffer Full)	A BF (Port A Buffer Full)
PC2	Input Port	Output Port	A $\overline{STB}$ (Port A Strobe)	A $\overline{STB}$ (Port A Strobe)
PC3	Input Port	Output Port	Output Port	B INTR (Port B Interrupt)
PC4	Input Port	Output Port	Output Port	B BF (Port B Buffer Full)
PC5	Input Port	Output Port	Output Port	B $\overline{STB}$ (Port B Strobe)

ALT1 and ALT2 correspond to simple input/output of Port C respectively. In ALT3 mode, $PC_0 - PC_2$ bits are used as control signals for port A, while pins $PC_3 - PC_5$ act as output pins. In ALT4 mode, $PC_0 - PC_2$ bits are used as control signals for port A, while $PC_3 - PC_5$ bits are used as control signals for port B.

9. Draw the status word format and discuss the same.

Ans. The status word format of 8155 is given below:

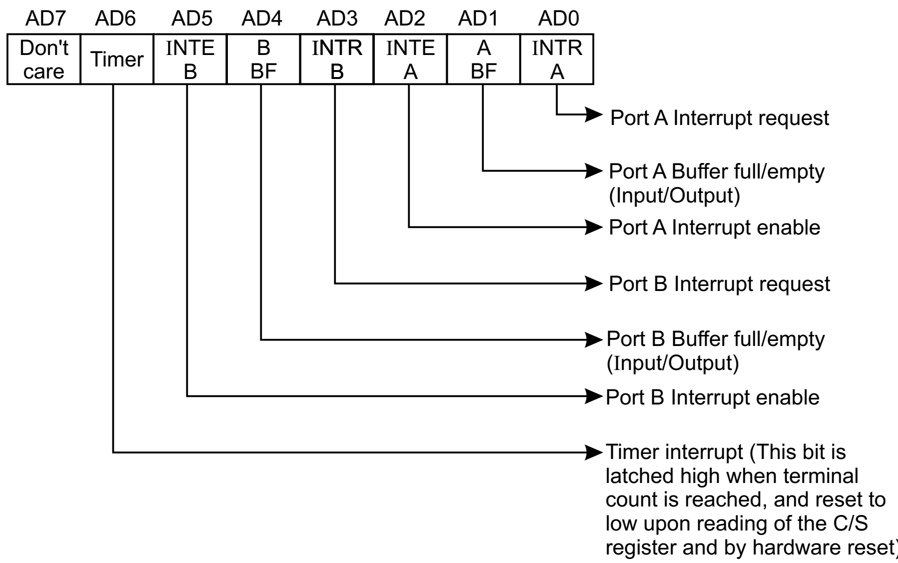


Fig. 9b.4: Status word format (*Source:* Intel Corporation)

It has seven latches. Bit D_7 is the 'don't care' bit. Bit D_6 contains the status of the timer. Bits $D_5 - D_3$ pertain to status of port B while bits $D_2 - D_0$ to that of Port A.

10. Discuss the timer section of 8155 and discuss its operating modes.

Ans. The timer section consists of two 8-bit registers. 14-bits of the two registers comprise to specify the count of the timer, which counts in a count-down manner. Contents of bits 6 and 7 of the most significant byte of the register decide the mode of operation of the counter. The following shows the timer register format. The timer section needs a 'TIMER IN' pulse, which is fed via pin 3 of 8155. A square wave or a pulse is obtained via pin 6 $\overline{\text{(TIMER OUT)}}$ when the terminal count (TC) is reached. The maximum and minimum values of the count down timer are $3FF_H$ and 002_H respectively. A single square wave or a continuous square wave or a single pulse on TC or a pulse on each TC (i.e., continuous pulses) are obtained, depending on the mode setting bits M_2 and M_1.

The following figure shows the nature of the outputs for the different modes.

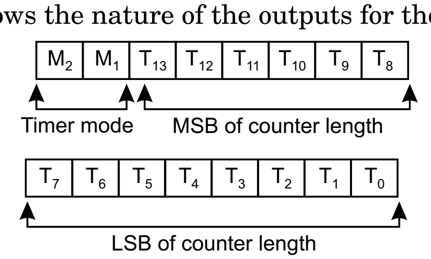


Fig. 9b.5: Timer registers format (*Source:* Intel Corporation)

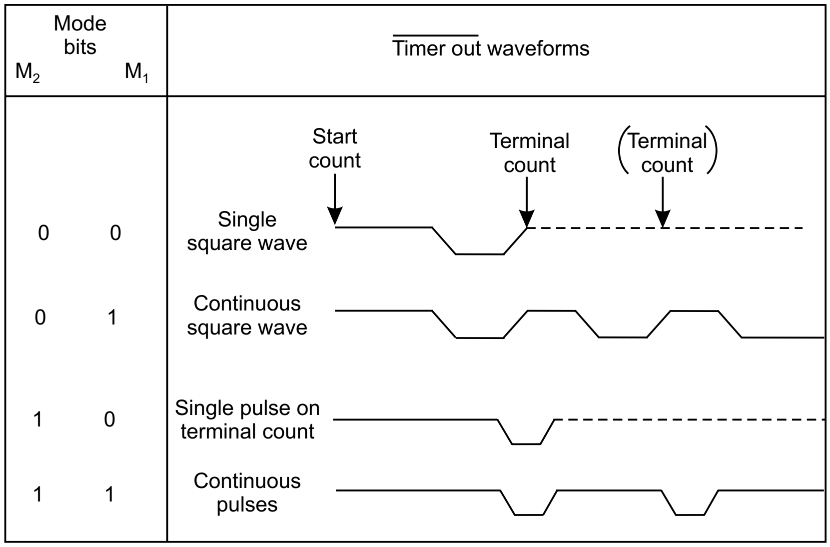


Fig. 9b.6: Timer modes and outputs (*Source:* Intel Corporation)

11. What happens when a high is applied on RESET ?

Ans. A high reset input resets the counter. To restart counting after resetting, a START command is required through the control register.

9c

8355/8755: Programmable I/O Ports with ROM/EPROM

1. Draw the pin connection diagram of 8355.

Ans. The pin connection diagram of 8355 is shown in Fig. 9c.1.

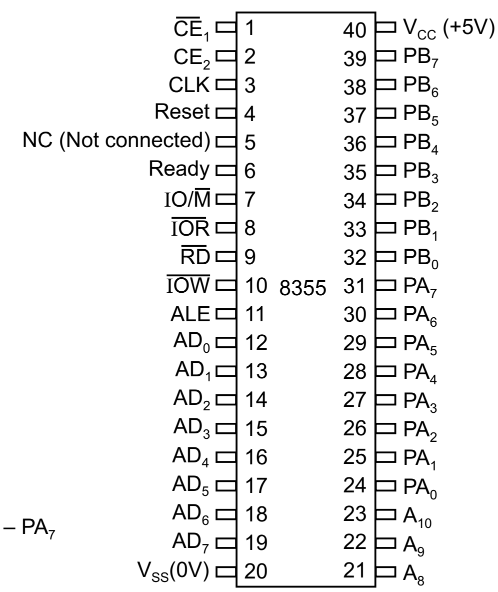


Fig. 9c.1: Pin diagram of 8355 (*Source:* Intel Corporation)

2. Draw the functional block diagram of 8355 and discuss.

Ans. The functional block diagram of 8355 is shown in Fig. 9c.2.

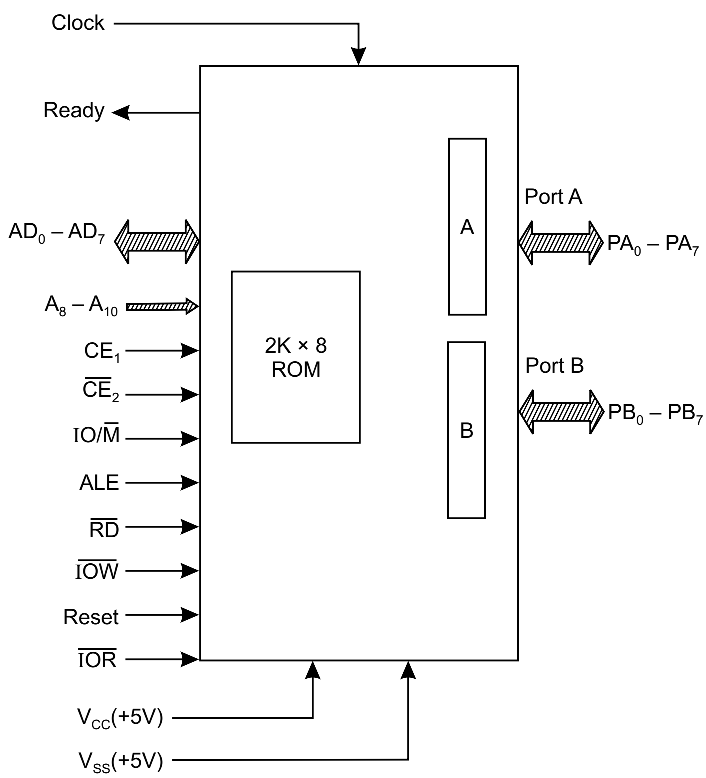


Fig. 9c.2: Functional diagram of 8355 (*Source:* Intel Corporation)

3. What is the difference between 8355 and 8755?

Ans. The 2 KB memory for 8355 is a ROM, while that for 8755, it is EPROM.

Again for 8355, there are two chip enable signals—CE_1 and $\overline{CE}_2$, while for 8755 this signal is designated as CE_2.

Both have two I/O Ports—each I/O line of either port can be programmed either as input or output.

4. What the DDR's do?

Ans. There are two internal control registers, called Data Direction Registers (DDRs)—both the registers are 1-byte in length and designated as DDR_A an DDR_B. Each bit in the two DDR registers control the corresponding bit in the I/O ports. For example bit D_0 of DDR_A controls D_0 bit of Port A and bit D_5 of DDR_B controls D_5 bit of Port B.

5. How the two ports and the two DDRs are selected?

Ans. The bits AD_1 and AD_0 controls/selects one out of the four of the above. This is like this:

AD_1	AD_0		Selected Port or DDR
0	0	⇒	Port A
0	1	⇒	Port B
1	0	⇒	DDR_A
1	1	⇒	DDR_B

The $IO/\overline{M}$ signal is to remain high during the above.

6. How the 2 KB ROM of 8355 is accessed?

Ans. The 2 KB ROM of 8355 is accessed by the $A_{10} – A_0$ latched address in conjunction with a low on $IO/\overline{M}$ signal.

9d

8279: Programmable Keyboard/Display Interface

1. Draw the pin diagram of 8279.

Ans. The pin diagram of 8279 is shown below:

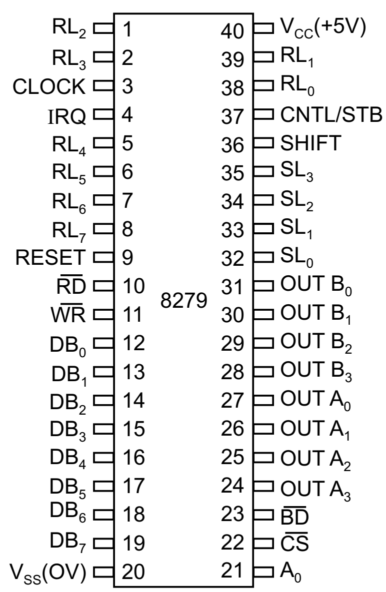


Fig. 9d.1: 8279 pin diagram
(*Source:* Intel Corporation)

2. Draw the functional block diagram of 8279 and elaborate on the different blocks.

Ans. The functional block diagram of 8279 is shown below:

The different functional blocks of 8279 are (*a*) a CPU interface, (*b*) a set of scan lines, (*c*) input lines for key data and (*d*) output lines for display data.

The CPU interface consists of 8-bit data bus along with $\overline{CS}$, $\overline{RD}$, $\overline{WR}$, CLK, RESET and IRQ lines. IRQ is an output line which becomes 1 (active) when key data exists in an internal RAM of 8279. This line is normally connected to one of the hardware interrupt

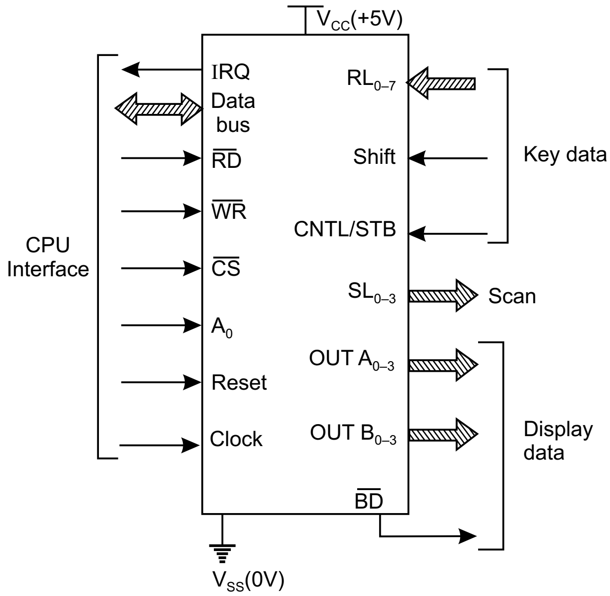


Fig. 9d.2: Functional block diagram of 8279
(*Source:* Intel Corporation)

lines of the CPU. A high on A_0 indicates that the signals in/out pertain to command/status while a low on A_0 indicate that they are data.

The scan lines (SL_{0-3}) along with the eight return lines (RL_{0-7}) can be used for construction of a keyboard matrix. SHIFT and CNTL/STB signals (both inputs) contribute the characteristics for individual keys.

The display output is available through A_{0-3} and B_{0-3} which can be used together as an 8-bit port. $\overline{BD}$ (output signal) is used for display blanking purposes.

3. What are the two most important functions performed by 8279?

Ans. The two most important functions performed by 8279 are as follows:

(*a*) It scans the keyboard, then detects the key press and transmits to the CPU information which corresponds to the particular key pressed.

(*b*) It puts out data received from the CPU, for use by the display devices.

4. What are the various input modes in which 8279 operate?

Ans. There are three input modes in which 8279 operates:

- Scanned Keyboard Mode
- Scanned Sensor Matrix Mode
- Strobed Input Mode.

5. How many character definitions are possible using 8279?

Ans. A maximum of 256 character definitions are possible using 8279.

6. When the CPU is actually involved for the scan and display functions to be realised?

Ans. For the above two functions to be realised, CPU involvement is required only when data is actually transmitted to or received from the CPU.

7. What are the modes in which the four scan lines can operate?

Ans. The four scan lines ($SL_0 - SL_3$) can be operated in two modes—encoded and decoded mode.

8. Discuss the encoded and decoded mode.

Ans. *Encoded mode:* Here 16 lines are generated using the 4 scan lines and a 4 to 16 *external* decoder, although the manufacturers recommend not to use the SL_3 line. Thus eight decoded scan lines are possible with $SL_0 - SL_2$ lines and a 3 to 8 decoder. These 8 lines, along with eight return lines ($RL_0 - RL_7$) can form a 8 × 8 keyboard matrix. Thus it leads to 64 different character definitions. With SHIFT and CONTROL input lines taken as two additional input lines, total character definitions possible = $64 \times 2^2 = 256$.

Decoded mode: Using the internal decoder present in 8279, SL_0–SL_3 lines are decoded. With SHIFT and CONTROL lines along with RL0 – RL7 lines, total character definition possible here is = 4 × 8 × 4 = 128.

9. Describe the Scanned Keyboard Mode.

Ans. Both encoded and decoded scan versions are applicable in this case. This mode can be divided into two ways.

- 2 key lockout
- N-key rollover

In this mode, the pressing of a key generates a unique 6-bit data (called 'position data') which is characteristic of the position of the key pressed. These 6-bits, along with CNTL and SHIFT form a 8-bit word, shown below. Of the position data $D_5 - D_0$, Scan bits correspond to $D_5 - D_3$ and Return bits correspond to $D_2 - D_0$. $D_5 - D_3$ bits correspond to the position of the row on which the key is pressed while $D_2 - D_0$ correspond to the position of the column on which the key is pressed. This 8-bit word gets stored in the RAM of 8279 (in FIFO order) and consequently the IRQ (interrupt request, an output line) line goes high. This IRQ line is connected to one of the hardware interrupt pins of the CPU. On recognition of the interrupt input by the CPU, the RAM in 8279 is read in FIFO form. Once this reading by CPU is over IRQ line of 8279 goes low but will become high if the RAM contains another data.

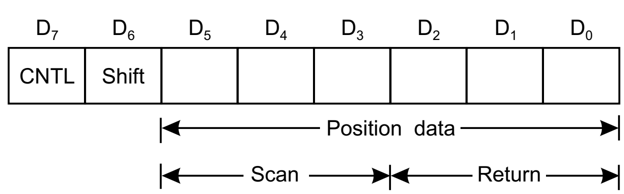


Fig. 9d.3: The scanned keyboard mode format

2 Key Lockout: In this 2 key lock out version of Scanned Keyboard Mode, when any key is pressed, it waits for next two scans to check whether any other key is pressed or not. Several possibilities do arise which need to be addressed separately.

(*a*) No other key press is detected. Then data corresponding to key press is taken to RAM in 8279 and IRQ output line goes into high state.
In case this internal RAM (of 8279) is already full, the keyed data is ignored and the error flag is set (= 1).

(*b*) If one or more additional key pressing occurs, no data entry into RAM is allowed. In this case two possibilities occur:

(*i*) If the first key (i.e., the key which was pressed first) is released ahead of others, then the key press is ignored.

(*ii*) If all the keys are released before the key first pressed, then data corresponding to first key pressed, is entered into RAM of 8279.

Another possibility is pressing of two keys within one debounce cycle (the time required for eliminating contact bounce effect is known as contact debounce time). In this case, no key is recognised. When one key is released, the other key that remains pressed is recognised as a single valid key depression.

N-Key Rollover: In this case, the debounce circuit waits for two scans after the first key press. It then checks whether key is still in the pressed condition or not. If the answer is yes, then the data corresponding to the key press is taken into RAM of 8279. No limit is there to the number of key presses. For simultaneous key presses, data are entered according to the order of key press.

If within a single debounce cycle, two keys are found pressed, the error flag is set and data entry into the RAM is prohibited. The error flag can be read from the FIFO STATUS word and can be cleared by a CLEAR command ($C_F = 1$).

10. Describe the Scanned Sensor Matrix Mode.

Ans. In the Scanned Sensor Matrix Mode of operation, the keys are arranged in the form of a matrix, with the scan lines ($SL_0 - SL_2$) forming the columns and return lines ($RL_0 - RL_7$) forming the rows. The open/closed condition of the key is stored in a RAM location. The size of the matrix be 8 × 8 or 4 × 8 for encoded and decoded scan lines respectively.

The data entering via the RL lines are admitted into eight columns of the sensor RAM—thus each RAM position corresponds to a specific switch position. Apart from switches, other logic circuit output lines can be connected to the RL lines.

11. Describe the Strobed Input Mode.

Ans. In this mode, data are placed on the return lines (RLs). The source of data may be an encoded keyboard or a switch matrix. The data so entering go to FIFO RAM and are accepted on the rising edge of a CNTL/STB pulse.

12. State the options available in the display mode.

Ans. The available options are:

- Display format—either left entry (also known as typewriter mode), or right entry (also known as calculator mode).
- Number of display characters: eight or sixteen.
- Organisation of characters—Single 8-bit or dual 4-bit type.

13. Discuss the Left Entry (Typewriter) Mode of Display format.

Ans. In the left entry (or typewriter) mode, the first entry goes to address 0, the second entry to address 1 and so on. The first entry goes to the left most display position. The second entry to the just right of the earlier one. Thus the 16th entry goes to 15th address position. It is to be remembered that the 17th entry goes to the RAM address 0 again, 18th entry goes to RAM address 1 etc, and is shown in Fig. 9d.4.

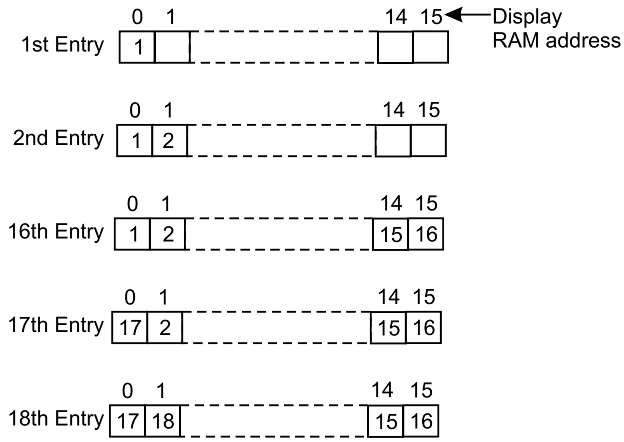


Fig. 9d.4: Left entry mode (Auto-increment)
(*Source:* Intel Corporation)

In this mode, data can be entered at any arbitrary RAM address position. Assuming a 8-position display, if a command 10010111 is inserted after the 2nd entry, then the next data will be displayed at 7th position. The explanation is like this: The most significant three bits 100 represent the code for WRITE display, the next bit, i.e., 1 is for auto-increment and the right most four bits i.e., 0111 (= 7) represent the position at which the next data will be filled in. This is shown in Fig. 9d.5.

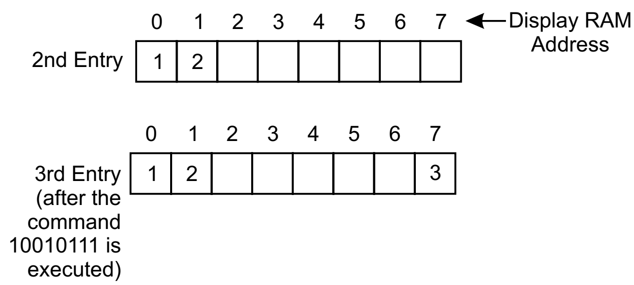


Fig. 9d.5

14. Discuss the Right Entry (Calculator) Mode of Display format.

Ans. In the right entry (calculator) mode, the characters are entered from the right most position. As characters are entered one after another, the present data occupies the right most position, just the earlier one occupies the left of the right most position etc. This is explained in Fig. 9d.6.

In this mode, no correspondence exists between RAM address and the display position.

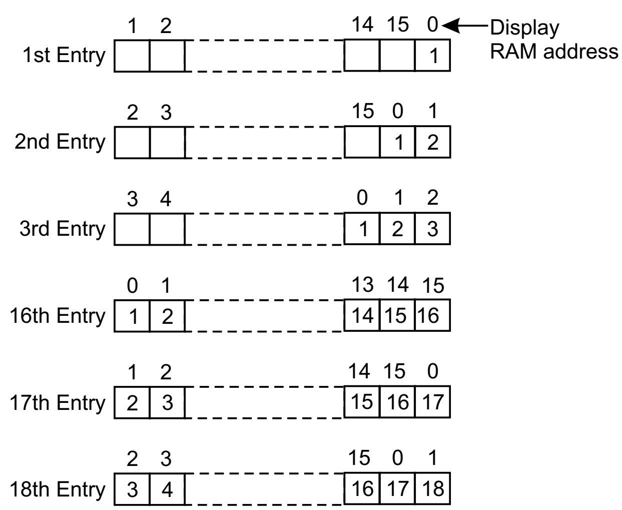


Fig. 9d.6: Right entry mode (Auto-Increment)
(*Source:* Intel Corporation)

15. What are the different types of software operations possible with 8279.

Ans. The following software operations are possible with 8279:

- Keyboard/display mode set
- Program clock
- Read FIFO/Sensor RAM
- Read Display RAM
- Write Display RAM
- Display Write Inhibit/Blanking
- Clear
- End Interrupt/Error Mode Set
- Status Word.

16. In how many ways data can be entered into a microprocessor?

Ans. There are three different ways of entering data into microprocessor—these are

- reading data from a DIP (on/off) switch.
- reading data from push-button keys.
- keys arranged in matrix form and read by software technique.

17. What is meant by contact bounce? How it is eliminated?

Ans. When an electromechanical switch is switched over from an off to on condition, the contact does not become firm on the first count. It loses contact and then makes it—this process repeats itself for a number of times before the contact is firmly placed. This occurs for a very small duration of time.

This thus leads to erroneous operation in digital circuits. This problem can be eliminated by a hardware circuit—called 'contact debouncers' or by software technique (by a delayed reading so that the transient period is over) in microprocessor based systems.

9e

Priority Interrupt Controller 8259

1. Draw the pin diagram of PIC 8259.

Ans. The following shows the pin details of PIC 8259.

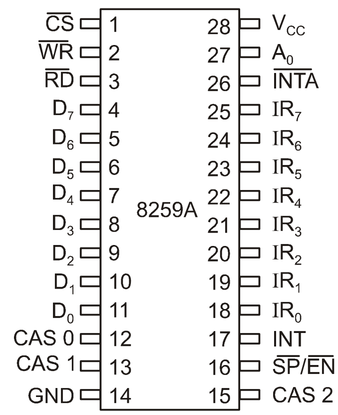


Fig. 9e.1: 8259 pin diagram (*Source:* Intel Corporation)

2. Draw the functional block diagram of PIC 8259.

Ans. The following shows the functional block diagram of 8259.

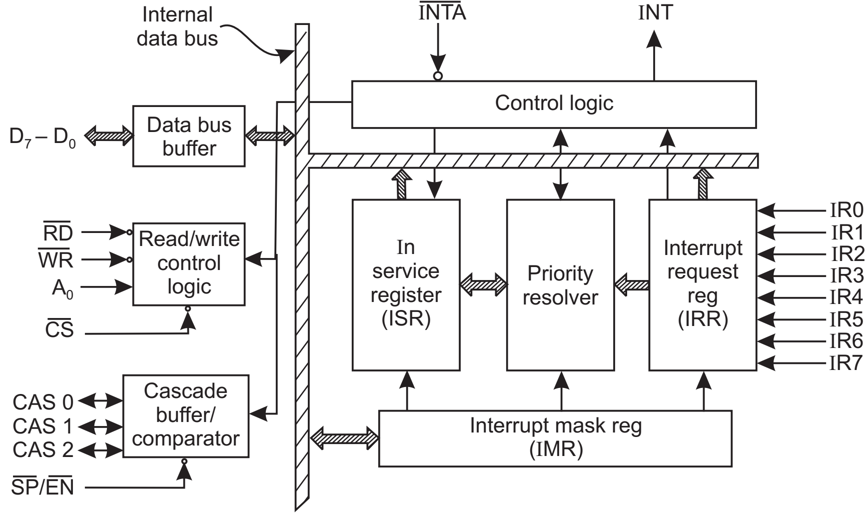


Fig. 9e.2: 8259 Functional block diagram (*Source:* Intel Corporation)

3. Draw the block schematic showing the interconnections between several I/O devices with PIC 8259, μP, RAM, ROM, etc.

Ans. The block schematic of the interconnections is shown below:

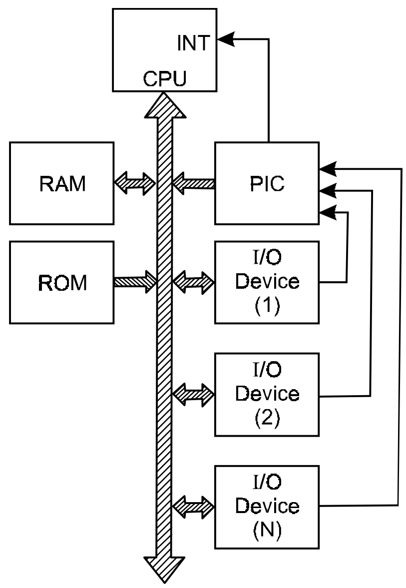


Fig. 9e.3: PIC in an interrupt-driven environment
(*Source:* Intel Corporation)

4. How many interrupt levels can be handled by 8259?

Ans. A single 8259 can handle up to 8 levels of interrupts.

Several 8259's can be cascaded to handle up to 64 levels of interrupts.

5. In how many interrupt modes can 8259 operate and which command word is utilised for this?

Ans. 8259 can be operated in the following categories of interrupt modes:

(*a*) fully nested mode
(*b*) rotating priority mode
(*c*) special mask mode
(*d*) polled mode

Operation Command Word (OCW) is used for 8259 to be operated in the above mentioned modes.

6. How the 8259 is programmed?

Ans. 8259 is programmed by a set of Initialisation Command Words (ICWs). Each 8259 attached to the system must be initialised through ICWs. In all there are four ICWs (ICW1 to ICW4).

7. What are the jobs performed by ICWs?

Ans. ICWs perform the following jobs:

- specifying the vectoring addresses for the individual interrupts.

- specifying single or cascaded mode of operation.
- level or edge triggering mode of operation.

8. What are the jobs performed by OCWs?

Ans. The operation command words (OCWs) are used to operate the PIC 8259 in various interrupt modes like fully nested mode, rotating priority mode, special mask mode and polled mode. The OCWs are also used for masking specific interrupts, status read operations, etc.

9. Describe how the PIC 8259 responds to interrupts?

Ans. PIC 8259 can accept a maximum of 8 interrupts from 8 different I/O devices, resolves the priority with regard to servicing the interrupts and issues an INT output signal, which is connected to INTR input pin (pin 10 of 8085). 8085, in its turn, issues an $\overline{\text{INTA}}$ signal via its pin 11, which is connected to $\overline{\text{INTA}}$ signal of 8259 (pin 26, an input pin for 8259). In response, 8259 puts out a CALL instruction code on the data bus. This is read and decoded by 8085, which then puts out two more $\overline{\text{INTA}}$s. This is done by 8085 to read the address where the Interrupt Service Subroutine (ISS) is written. Now the PIC 8259 puts out the address of this ISS on the data bus which is eventually read by 8085 and the program jumps to the ISS address as was previously programmed by 8259.

10. What are the different functional blocks in 8259?

Ans. PIC 8259 has four different functional blocks viz.,

(*i*) Interrupt and Control logic block
(*ii*) Data bus buffer
(*iii*) Read/Write control logic block and
(*iv*) Cascade buffer/comparator section.

11. What the Interrupt and Control Logic Section consist of?

Ans. This section consists of (*a*) Interrupt Request Register (IRR), (*b*) In Service Register (ISR), (*c*) Priority Resolver, (*d*) Interrupt Mask Register (IMR), (*e*) Control Logic Block.

12. Write down the sequence of operations for programming 8259.

Ans. 8259 is programmed by issuing initialisation command words and operation command words. Initialisation command words are issued in a sequence. The following is the algorithm for initialising 8259.

1. Write ICW1.
2. Write ICW2.
3. If not in the cascade mode of operation, Go To Step 5.
4. Write ICW3.
5. IF ICW4 is not needed, then Go To Step 7.
6. Write ICW4.
7. Ready to accept interrupt sequence.

The flowchart for the above is shown in Fig. 9e.4.

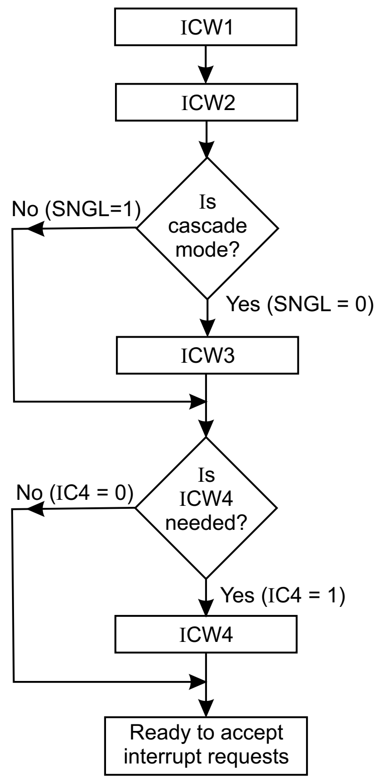


Fig. 9e.4: 8259 initialisation flow chart
(*Source:* Intel Corporation)

13. Write down the main features of 8259.

Ans. The main features of 8259 are as follows:

1. A single 8259 can handle 8 vectored priority interrupts.
2. 9 numbers of 8259 can be cascaded to have 64 levels of vectored priority interrupts in a μC system.
3. The priority modes can be changed or reconfigured dynamically at any time during the main program.
4. It can be operated in various interrupt modes—fully nested, rotating priority, special mask and polled.
5. 8259 can be used with either 8080/8085 or 8086/8088 microprocessor.
6. 8259 supports both edge and level triggered mode of interrupts.
7. The CALL address can be programmed to have a spacing of either 4 or 8 memory locations.
8. The data bus is buffered.
9. The AEOI (Automatic End Of Interrupt) can be programmed.

14. Write down the functions of IR0–IR7 pins.

Ans. There are 8 interrupt input lines from external devices with IR0 having the highest and IR7 the lowest priority. These interrupt requests are acknowledged by 8259 by (*a*) raising the corresponding IR input (i.e., any one of IR0 to IR7) from L to H and holding it high until acknowledged or (*b*) just by a high level.

15. What is the function of $\overline{SP}/\overline{EN}$ pin?

Ans. This is slave program/enable buffer pin of 8259. It has dual functions:

(*a*) In the buffered mode it is used as an output to control the buffer transreceivers (EN). It acts as an output pin to enable the data bus buffer of the system.

(*b*) In the non-buffered mode, it is used as an input pin to designate the 8259 to operate as a master (SP = 1) or slave (SP = 0).

The buffered or non-buffered mode of operation is determined at the time of initialisation of 8259 via ICW4.

16. Describe the functioning of the pins CAS0 – CAS2.

Ans. In a multiple 8259 structure, these three CAS lines form a private 8259 bus.

For a master 8259, these three pins act as output pins, but act as input pins for a slave 8259.

In a multiple 8259 structure, the master 8259 accepts interrupt requests from slave 8259. The master 8259 then generates a CALL opcode in response to the first $\overline{INTA}$. The slave 8259 provides the vectoring address. The master, via its three output pins (CAS0 – CAS2), gives out a code to identify the slave 8259. The slave 8259's, connected to the system, each accepts this code from master 8259 and compare it with the code assigned to it during initialisation. The slave, so identified, then puts out the address of the interrupt service subroutine on to DATA BUS during the second and third $\overline{INTA}$ pulses from the CPU.

The identification of this master and slave 8259's is done by the cascade buffer/comparator block.

17. Discuss the vector data formats when 8085 is interrupted via INTR.

Ans. There are eight interrupt levels (IR0 – IR7) that generate CALL to eight equally spaced locations in the memory. These eight locations can be programmed to be spaced at an interval of either 4 or 8 memory locations.

The first interrupt vector byte released in response to the first $\overline{INTA}$ is shown in Fig. 9e.5. It is equivalent to the opcode for CALL instruction.

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CALL CODE	1	1	0	0	1	1	0	1

Fig. 9e.5: First interrupt vector byte
(*Source:* Intel Corporation)

In response to the second $\overline{INTA}$, the second interrupt vector byte released is shown in figure 9e.6. which shows interval spacings of either 4 or 8.

When the interval spacing is 4, 8259 inserts D_0 – D_4 bits automatically as per the levels of interrupts (i.e., IR0 – IR7), while D_5 – D_7 bits are specified during programming of 8259 through ICW1. Again for an interval of 8 between consecutive memory locations, D_0 – D_5 bits are inserted by 8259 automatically, while D_6 – D_7 bits are specified during programming of 8259 through ICW1.

IR	Interval = 4							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	A5	1	1	1	0	0
6	A7	A6	A5	1	1	0	0	0
4	A7	A6	A5	1	0	1	0	0
5	A7	A6	A5	1	0	0	0	0
3	A7	A6	A5	0	1	1	0	0
2	A7	A6	A5	0	1	0	0	0
1	A7	A6	A5	0	0	1	0	0
0	A7	A6	A5	0	0	0	0	0

IR	Interval = 8							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	1	1	1	0	0	0
6	A7	A6	1	1	0	0	0	0
5	A7	A6	1	0	1	0	0	0
4	A7	A6	1	0	0	0	0	0
3	A7	A6	0	1	1	0	0	0
2	A7	A6	0	1	0	0	0	0
1	A7	A6	0	0	1	0	0	0
0	A7	A6	0	0	0	0	0	0

Fig. 9e.6: Second interrupt vector byte
(*Source:* Intel Corporation)

The following figure shows the format of the byte released by 8259 in response to the third $\overline{\text{INTA}}$.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8

Fig. 9e.7: Third interrupt vector byte
(*Source:* Intel Corporation)

This byte is completely programmable and is specified by ICW2.

18. Discuss the Initialisation Command Word 1 (ICW1).

Ans. Whenever a Write Command is received with $A_0 = 0$ and $D_4 = 1$, it is interpreted by 8259 to be an initialisation command word ICW1. During ICW1, the following occur:

(*i*) The edge sense circuit is reset, which means that following initialisation, an interrupt request (IR) input must make a L to H transition to generate an interrupt.
(*ii*) The interrupt mask register (IMR) is cleared, i.e., all interrupts are now disabled.
(*iii*) IR7 input is assigned priority 7 (lowest).
(*iv*) The slave mode address is set to 7.
(*v*) The special mask mode is cleared and Status Read is set to IRR.
(*vi*) If D_0 bit in ICW1 (IC4) is set to 0, then all functions selected in ICW4 are set to zero.

The format of the byte to be followed for ICW1 is shown below:

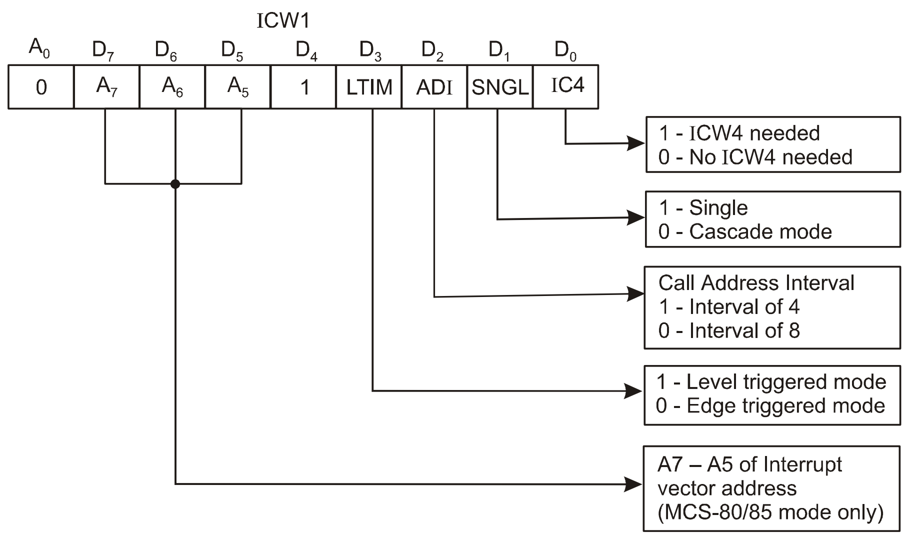


Fig. 9e.8: Initialisation command word 1 (*Source:* Intel Corporation)

- **Bit D_0 (IC4):** It indicates whether ICW4 is needed or not. If it is '1', then ICW4 is needed and if '0', then ICW4 is not needed.
- **Bit D_1 (SNGL):** If this bit is '0', then only one 8259 is in the system and if it is '1' then additional 8259's are there in the system.
- **Bit D_2 (ADI):** ADI stands for 'address interval'. If this bit is '0' then call address interval is 8 and if '1' then call address interval becomes 4.
- **Bit D_3 (LTIM):** This bit determines recognition of the interrupts either in level triggered or edge triggered mode. If this bit = '0', then it is edge triggered mode and if this bit is = '1' then the input interrupts will be recognised if they are in the level triggered mode.
- **D_5 – D_7:** These are $A_5 - A_7$ bits as shown under ICW1. For an interval spacing of 4, $A_0 - A_4$ bits are automatically inserted by 8259 while $A_0 - A_5$ are inserted automatically for an interval of 8. $A_5 - A_7$ bits are programmable as set by the bits $D_5 - D_7$ of ICW1.

19. Discuss the Initialisation Command Word 2 (ICW2).

Ans. The initialisation command word 2 i.e., ICW2 is shown below:

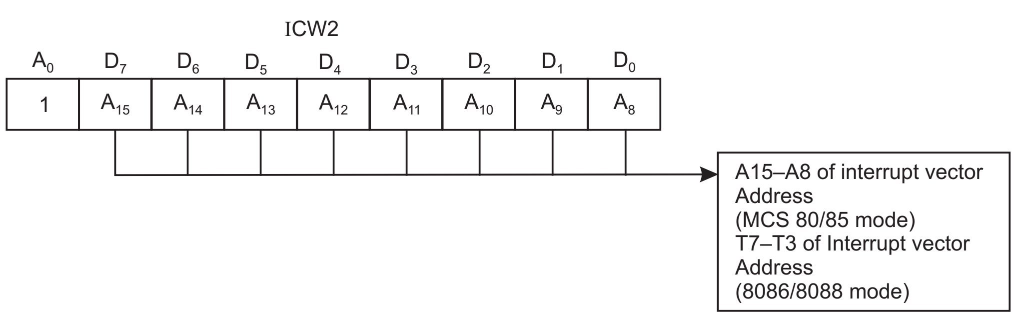


Fig. 9e.9: Initialisation command word 2 (*Source:* Intel Corporation)

If A_0 = 1 of the write command word is issued to 8259, following ICW1, it is interpreted as ICW2. ICW2 is used to load the high order byte of the interrupt vector address of all the interrupts. This byte is common for all the interrupts.

20. Discuss the Initialisation Command Word 3 (ICW3).

Ans. ICW3 can have two modes of operations: Master Mode ICW3 and Slave Mode ICW3. ICW3 is required only if several 8259's are used in the μC system in a cascaded form.

The format of the byte (both for Master Mode ICW3 and Slave Mode ICW3) are shown below:

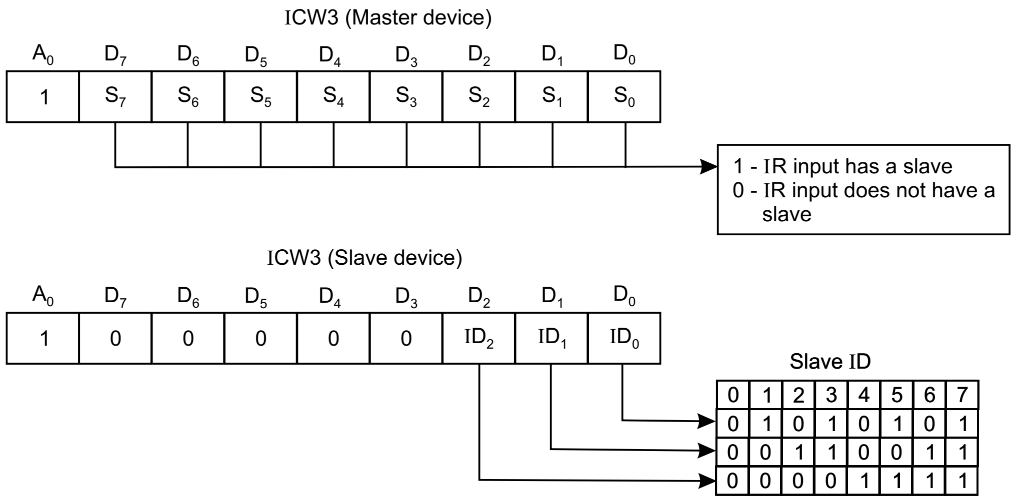


Fig. 9e.10: Initialisation command word 3
(*Source:* Intel Corporation)

Master Mode ICW3: For a 8259 to be treated as a master, we must have $\overline{SP}/\overline{EN}$ pin = 1 in a non-buffered environment and M/S = 1 in ICW4 in a buffered environment.

Then each bit in ICW3 is used to indicate to the master whether it has a slave 8259 attached to it on its corresponding interrupt request (IR) input pin. A '1' indicates the presence of a slave 8259 corresponding to that input and a '0' indicates the absence of a slave.

If now a particular slave 8259 raises its INTR output, then the master generates a CALL instruction opcode and puts out the slave identification number on its output CAS0 – CAS2 lines. This number goes to the slave 8259's via their CAS0 – CAS2 pins (these pins act as input pins for slave 8259's). Thus the number is compared with the individual slave identification number loaded during initialisation. The slave which initially placed the INTR output is thus identified and hence it releases the vector address during the second and third $\overline{INTA}$ cycles.

Slave Mode ICW3: For a 8259 to be treated as a slave, we must have $\overline{SP}/\overline{EN}$ pin = 0 in a non-buffered environment and M/S = 0 in a ICW4 in a buffered environment.

Bits $D_0 - D_2$ of ICW3 (in Slave Mode) assign the slave identification code (Slave ID). The slave ID is equivalent to the master IR input to which the INTR output of the slave is connected. The slave ID compares this number with its own CAS0 – CAS2 inputs so as to release the address vector.

21. Describe the Initialisation Command Word 4 (ICW4).

Ans. The format of ICW4 is shown below. ICW4 is loaded only if D_0 bit of ICW1 (IC4) is set.

As shown in Fig. 9e.11, ICW4 is loaded only if D_0 bit of ICW1 (IC4) is set. The format of ICW4 is shown below.

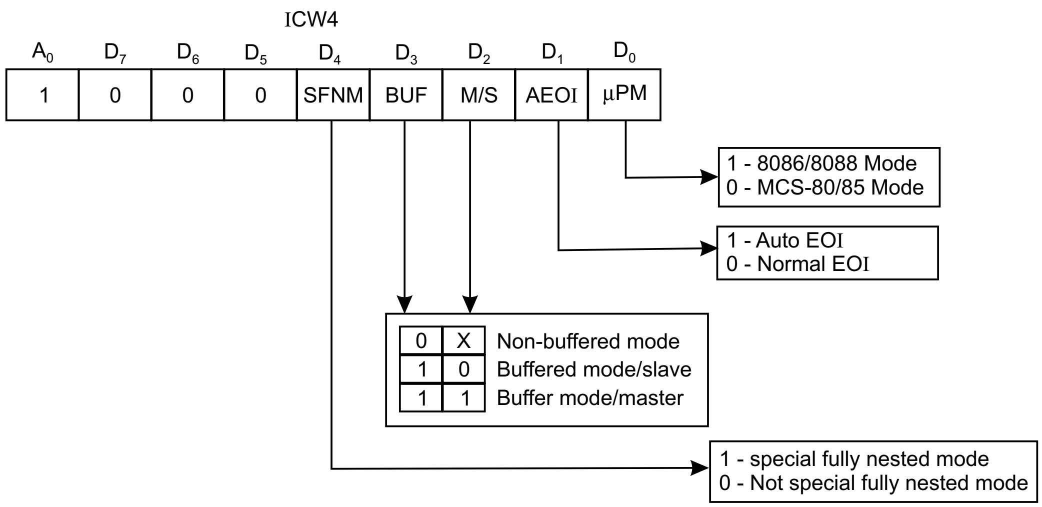


Fig. 9e.11: Initialisation command word 4
(*Source:* Intel Corporation)

The bit positions D_0 to D_4 are now explained:

µPM: This corresponds to D_0 bit position and differentiates between 8086/8088 mode and MCS-80/85 mode.

AEOI: It stands for 'automatic end of interrupt'. If AEOI bit (bit D_1) = 1, then it is in auto EOI mode and if it is = 0, it is in normal EOI mode.

M/S: In the buffered mode, if M/S = 1, then 8259 is initialised as a master and if M/S = 0, then 8259 acts as a slave.

In the non-buffered mode, M/S pin has no significance. In this case, the characteristics of 8259 (i.e., whether 8259 is a master or a slave) is determined by $\overline{SP}/\overline{EN}$ pin.

BUF: This bit position (D_3 bit) determines buffered/non-buffered mode of operation. If BUF = 1, then it is buffered mode of operation and the $\overline{SP}/\overline{EN}$ pin is used as an output to enable the data bus buffer of the system.

SFNM: This stands for Special Fully Nested Mode (bit D_4). If SFNM = 1, then this mode is programmed.

22. Describe the Operation Command Words (OCWs).

Ans. There are three OCWs. These OCWs may be required to change the manner in which the interrupts are to be processed. For this to be achieved, the OCWs may be loaded any time after initialisation is over.

The format for the OCW1 is shown in Fig. 9e.12.

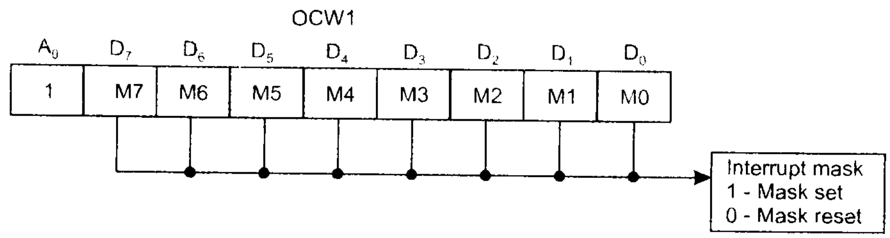


Fig. 9e.12: Format of operation command word 1
(*Source:* Intel Corporation)

OCW1: It is used to enable/disable a particular interrupt request by programming the Interrupt Mask Register (IMR). If M = 1, then the corresponding interrupt is masked and M = 0 indicates its unmasked condition.

A write command with $A_0 = 1$ is interpreted as OCW1, and written after ICW2.

OCW2: The format for OCW2 is shown below:

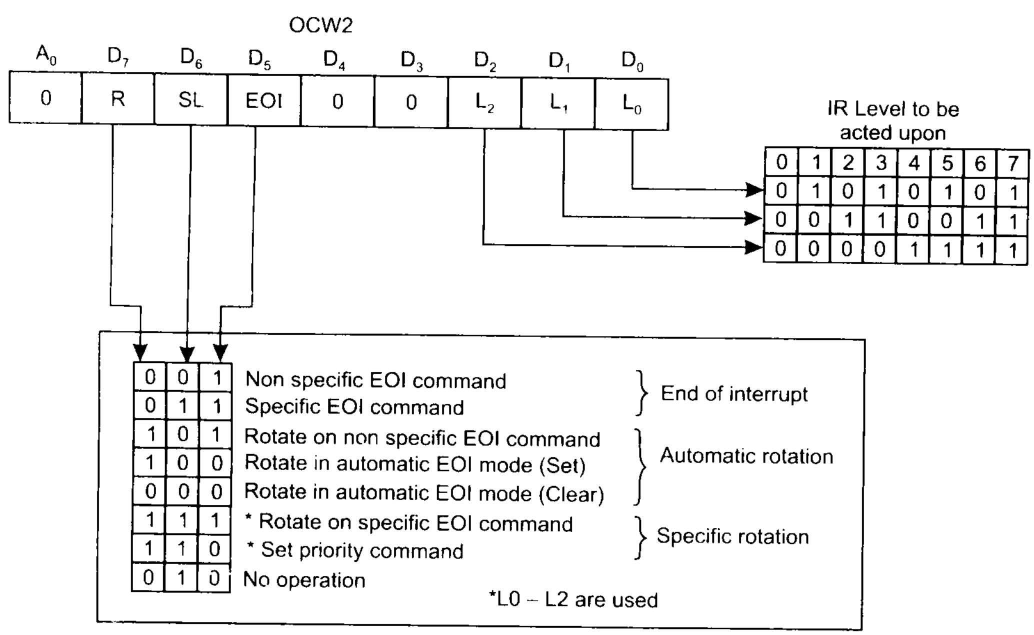


Fig. 9e.13: Format of operation command word 2
(*Source:* Intel Corporation)

A write command with $A_0 = 0$ and $D_4 D_3 = 00$ is interpreted as OCW2. 'R' and 'SL' stand for Rotate and Select Level respectively. The bits corresponding to R, SL and EOI control the Rotate and End of Interrupt modes. Bits $L_2 - L_0$ specify the interrupt level which is to be acted upon when SL is in active condition.

OCW3: The jobs performed by OCW3 are as follows:

(*a*) to read the status of registers.

(b) set/reset the Special Mask and Polled modes.

The format of OCW3 is shown Fig. 9e.14.

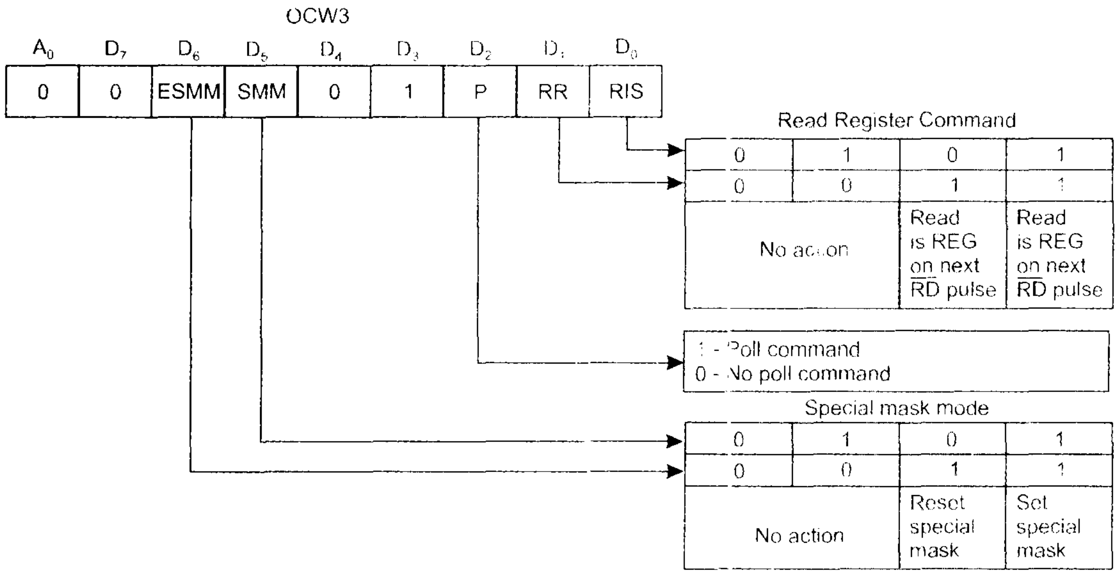


Fig. 9e.14: Format of operation command word 3
(*Source:* Intel Corporation)

23. Describe the FNM (Fully Nested Mode).

Ans. This mode is auto-set after initialisation is over—i.e., it is a default mode. This mode can only be changed through Operation Command Words (OCWs).

IR0 is assigned the highest priority (priority 0) and IR7 the lowest priority (priority 7). When 8259 acknowledges an interrupt request via its INTR pin, it finds out the highest priority and the corresponding bit in the In Service Register (ISR) is set.

The resetting of this bit in ISR can occur in one of the following ways:

(a) When the CPU issues an EOI (End of Interrupt) through OCW before coming out of ISR.

(b) If AEOI (Automatic End of Request) mode is set in ICW4 during initialisation, the corresponding ISR bit is automatically reset—it occurs on the trailing edge of the last (third) $\overline{\text{INTA}}$ pulse.

When a particular ISR bit is set, (a) all lower level interrupts and (b) same level interrupts remain in the inhibited condition.

But a higher level interrupt will force the 8259 to generate an INTR, but the same will be acknowledged if the Interrupt Enable F/F has already been enabled via software while the interrupt service subroutine is in progress.

24. Describe the EOI command.

Ans. This command stands for End of Interrupt (EOI).

The interrupt service bit (that is being currently serviced) can be reset by an End of Interrupt Command. This is issued by the CPU, usually just before coming out of the interrupt service routine.

There are two ways in which the EOI can be exerted—in the Fully Nested Mode (FNM) and non FNM.

In the FNM, a *non-specific* EOI command is issued by the CPU. This is an OUT instruction by the CPU to 8259. This is derived by $A_0 = 0$, $D_7\ D_6\ D_5\ D_4\ D_3 = 00100$ with $D_2\ D_1\ D_0$ can have any value. This is apparent from the format of the operation command word 2 i.e., OCW2. This OUT instruction is issued by CPU before exiting from the interrupt service subroutine. On receiving this instruction, 8259 resets the highest level of interrupt (i.e., the current one that which is being serviced).

In non FNM, 8259 cannot determine the last interrupt acknowledged. Thus in this case, the CPU will have to issue a *specific* EOI command signalling out the specific interrupt service bit that is to be resetted. This is done under OCW2 with $A_0 = 0$, $D_7\ D_6\ D_5\ D_4\ D_3 = 01100$ and $D_2 - D_0$ specifies the level on which the EOI command is to act.

25. In the cascade mode, how many EOI commands are to be issued?

Ans. In such a case, two EOI commands must be issued—one for the master and one for the slave.

26. Describe the AEOI command.

Ans. This mode can only be used for a master 8259 and is set by ICW4. 8259 performs a non-specific EOI on the trailing edge of the third (i.e., last) $\overline{\text{INTA}}$ pulse.

27. Explain Special Fully Nested Mode (SFNM).

Ans. In the cascaded mode of operation, if a slave receives a higher priority interrupt request than one which is in service (through the same slave), it would not be recognised by the master. This is because the master ISR bit is already in the set condition, thereby it ignores all requests of equal or lower priority. The higher priority interrupt won't be serviced until after the master ISR bit is reset by an EOI command. This is most likely to happen after the completion of the lower priority routine.

This is where the SFNM comes into. It is meant only for the master and done during master initialisation (through ICW4). In this mode, the master will ignore interrupt requests of lower priority, but responds to requests of equal or higher priority.

The following are the differences between FNM and SFNM:

(*a*) In SFNM, the slave is allowed to place an interrupt request (of higher priority than the one currently being serviced). The master recognises this higher level interrupt, which in its turn places this interrupt request to the CPU.

(*b*) In SFNM, the software must determine if any other slave interrupts are pending before issuing an EOI command to the slave and then reading its ISR (In Service Register). If the ISR contains all zeroes, then no interrupt from the slave is in service and an EOI command can be sent to the master. If the ISR is not all zeros, an EOI command should not be sent to the master. Clearing the master ISR bit with an EOI command while there are still slave interrupts in service would allow lower priority interrupt to be recognised by the master.

28. Mention the types of Rotating Priority Mode of interrupt.

Ans. The Rotating Priority Mode can be set in

(*a*) Automatic Rotation

(*b*) Specific Rotation.

29. Discuss the Automatic Rotation.

Ans. In situations where several communicating channels are connected to µC system, all the channels should be accorded equal priority in sharing information with the µC.

Thus when a peripheral is serviced, all other equal priority peripherals should be given a chance to be serviced before the original peripheral is serviced a second time around. This is accomplished by automatically assigning a peripheral the lowest priority after being serviced. Thus a device, presently being serviced, would have to wait until all other devices are serviced.

Automatic rotation is of two types:

(*a*) Rotate on non-specific EOI Command.

(*b*) Rotate on automatic EOI Mode.

Rotate on non-specific EOI Command: When the rotate on NSEOI command is issued, the highest ISR bit is reset and the corresponding IR level is assigned the lowest priority.

Let IR0 has the highest and IR7 the lowest priority. Let also that IR6 and IR4 are in service with IR4 accorded the highest priority. Bit 4 in the ISR is reset when a NSEOI command is executed. After this, IR4 becomes the lowest priority and IR5 becomes the highest priority. The situations are explained in the following figure with the left side indicating the situation before the command is executed and the right side after the command execution.

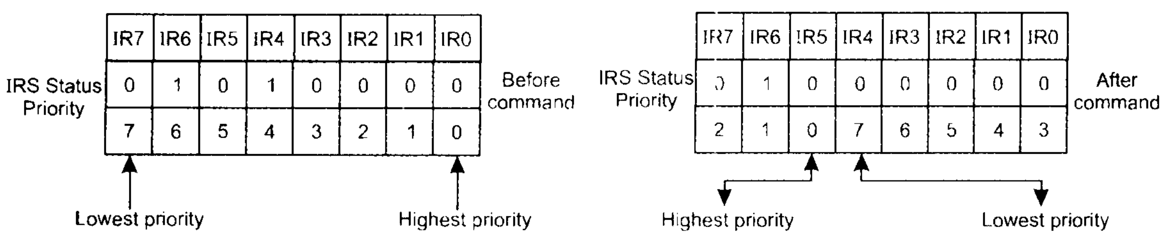


Fig. 9e.15: Rotation on non-specific EOI command

Rotate in Automatic EOI Mode: This mode works much like the rotate on NSEOI command. The main difference between the two lies in the priority routine done automatically after the last $\overline{\text{INTA}}$ pulse of an interrupt request. To enter or exit from this mode, a rotate-in-automatic EOI Set Command and rotate-in-automatic EOI Clear Command is provided.

30. Explain Specific Rotation.

Ans. In this mode, the lowest priority can be assigned to any of the IR levels (between 0 and 7) as specified by OCW2.

This mode is set by CPU by issuing an OUT instruction in the following manner:

$A_0 = 0$

$D_7\,D_6\,D_5\,D_4\,D_3 = 1\,1\,0\,0\,0$

$D_2\,D_1\,D_0$ bits specify which interrupt level is to be accorded lowest priority. This mode is independent of EOI command.

31. Describe Special Mask Mode.

Ans. The special mask mode enables interrupts from all levels except the level presently in service. This is done by masking the level that is in service and then issuing the special mask mode command. Once the special mask mode is set, it remains in effect until reset.

The Special Mask Mode can be set by making ESSM and SMM bits '1' in OCW3. When a mask bit is set in OCW1, all further interrupts at that level are inhibited, while interrupts on all other levels that have not been masked by OCW1 (both lower and higher) are enabled. Thus it is possible to selectively enable interrupts by programming the mask register.

The special mask mode is cleared by loading OCW3 as follows:

ESSM = 1

and SMM = 0

32. Describe the Polled Mode interrupt scheme.

Ans. In this mode, the interrupting devices seeking services from 8085 are polled one after another to detect which device has sought for interrupt request. The INT output of 8259 is either not connected to INTR input of 8085 or the interrupts are disabled by software means.

Bit D_2 (i.e., P) in OCW3 is set to '1' in the Polled mode. 8259 is then read by masking its $\overline{RD}$ and $\overline{CS}$ pins '0'. The ISR bit is set corresponding to the highest level interrupt in the IRR. A byte is put out on the data bus as shown below:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	–	–	–	–	W_2	W_1	W_0

Fig. 9e.16: Polled mode output word
(*Source:* Intel Corporation)

If $D_7 = 1$, then it implies that an interrupt needs servicing, while $D_2 - D_0$ bits (i.e., $W_2 - W_0$) correspond to the highest priority interrupt level which is requesting service.

Since INTR line is not in use in the polled mode, hence more than one 8259 may be connected in the master mode. Hence it is possible to have more than sixty four levels of interrupts in this mode.

33. On which registers status read operations can be done?

Ans. Status read operations can be done on Interrupt Request Register (IRR), In-Service Register (ISR) and Interrupt Mask Register (IMR).

34. How IRR status read operation is done?

Ans. A particular 8259 can be set up for an Interrupt Request Register (IRR) read operation by inserting in OCW3 RR (read register) = 1 and RIS (Read ISR) = 0. Following this, $\overline{RD}$ = 0 and $\overline{CS}$ = 0 are made on the 8259, which thereby puts out the contents of IRR on the data bus.

In the non-polled mode of 8259, A_0 should be made '0' so as to put out the contents of IRR status word after the IRR has been set for status read operation.

35. Discuss the ISR status read operation.

Ans. A particular 8259 can be set up for an In-Service Register (ISR) read operation by inserting in OCW3 RR = 1 and RIS = 1. Following this $\overline{RD}$ = 0 and $\overline{CS}$ = 0 are made on the 8259,

which thereby puts out the contents of ISR on the data bus.

In the non-polled mode of 8259, A_0 should be made '0' so as to put out the contents of ISR status word after the ISR has been set for status read operation.

36. How IMR status read operation is done?

Ans. An Interrupt Mask Register (IMR) status read operation on a 8259 is done with $A_0 = 1$, $\overline{RD} = 0$, $\overline{CS} = 0$. This causes the contents of IMR to be put out on the data bus. For an IMR status read operation, OCW3 is not needed.

37. Which status read operation is performed by default after initialisation of 8259?

Ans. The default status read operation is the IRR status read, after the initialisation of 8259.

38. Mention when status read operation is not possible.

Ans. When OCW3 is set in the polled mode with P = 1 and RR = 1, status read operation is not possible.

39. Discuss the Default IR7 routine.

Ans. An interrupt via the INTR input will be treated as a valid interrupt if it remains high until after the falling edge of the first $\overline{INTA}$ pulse. Then a valid IR7 input occurs resulting in a *normal* CALL to the IR7 routine. If it is otherwise, a *default* CALL to the IR7 routine will be generated.

A normal IR7 operation sets the ISR bit while a default IR7 operation does not do so. For this IR7 is normally used for RET instructions. If IR7 is to be utilised for other purposes, then the default IR7 operation is first to be checked. This is done by a status read operation of ISR at the beginning of interrupt service routine for IR7. If after this read operation, IR7 input = 1, then it is a valid interrupt, otherwise not.

40. Distinguish between NSEOI, SEOI and AEOI.

Ans. These three stand for non-specific end of interrupt, specific end of interrupt and automatic end of interrupt.

An NSEOI command sent from the µP lets the 8259 know when a service routine has been completed, but without any specification of its *exact* interrupt level. The 8259 determines the interrupt level (the highest priority interrupt in service) and resets the correct bit in the ISR.

The NSEOI is best suited when servicing is always at the highest priority level. When 8259 receives a NSEOI command, it simply resets the highest priority ISR bit.

The main advantage of this mode is that it is not necessary to specify the IR level. NSEOI command is not suited for the following two cases:

(*a*) Using a set priority command within an interrupt service routine.
(*b*) Using a Special Mask Mode.

A SEOI command sent from µP to 8259 lets it know when a service routine of a *particular* interrupt level is completed. A SEOI command resets a specific (particular) ISR bit—thus any one of the eight IR levels can be specified.

A SEOI command is needed when 8259 is unable to determine the IR level.

A SEOI command is best suited for situations in which priorities of the interrupt levels are changed during an interrupt routine (Specific Rotation).

The AEOI mode scores over the EOI modes in that no command has to be issued in

AEOI mode. Thus AEOI mode greatly simplifies programming and lowers code requirements within interrupt routines.

AEOI mode should be used continuously because the ISR bit of a routine presently in service is reset right after its acknowledgement. It thus leaves behind nothing in the ISR about which particular bit is being serviced. If any interrupt request occurs during this time, it will be serviced (provided all interrupts are enabled) regardless of its priority—whether low or high.

Another peculiar problem—called 'over nesting' may happen in this case. It occurs when an IR input keeps as interrupting its own routine. It results in unnecessary stack pushes which could fill up the entire stack in a worst case situation.

41. Describe how several PICs are cascaded together.

Ans. Several PICs can be cascaded together—in all 9-with one PIC acting as the master and the rest eight as slaves and shown in Fig. 9e.17. $\overline{SP}$ pin of the master is connected to V_{cc}, whereas for slave PICs, their $\overline{SP}$ pins are connected to ground. The INT outputs of the slave PICs are connected to one of IR0-IR7 pins of the master. The registers within the PICs are allocated separate addresses by using separate $\overline{CS}$ signals. Initialisation of each of the PICs are done separately.

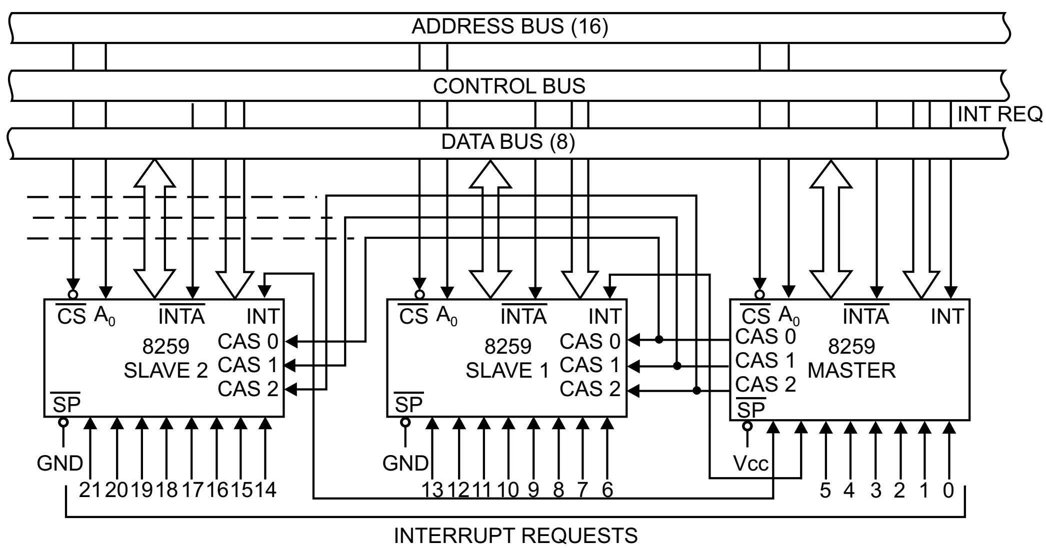


Fig. 9e. 17: Cascading of 8259 PICs

When an interrupt comes, it activates one of the IR input lines of a slave PIC and this in turn activates one of the IR lines of the master (via the INT output pin of the slave PIC which has been interrupted). This in turn interrupts the CPU. The INTA output from the master enables the corresponding CAS0-CAS2 lines of the slave PIC—this releases the vector address of the data bus in the second and third INTA cycles.

9f

Programmable DMA Controller (DMAC) 8257

1. Draw the pin diagram of 8257.

Ans. The following figure gives the pin connection diagram of 8257.

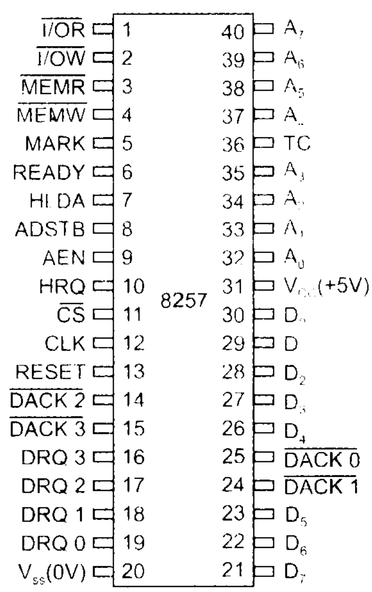


Fig. 9f.1: 8257 pin diagram (*Source:* Intel Corporation)

2. Draw the functional block diagram of 8257.

Ans. The following figure shows the functional block diagram of the DMAC 8257.

As is apparent from the figure, it consists of eight blocks: data bus buffer, read/write block, control logic and mode set register, priority resolver and four channel blocks.

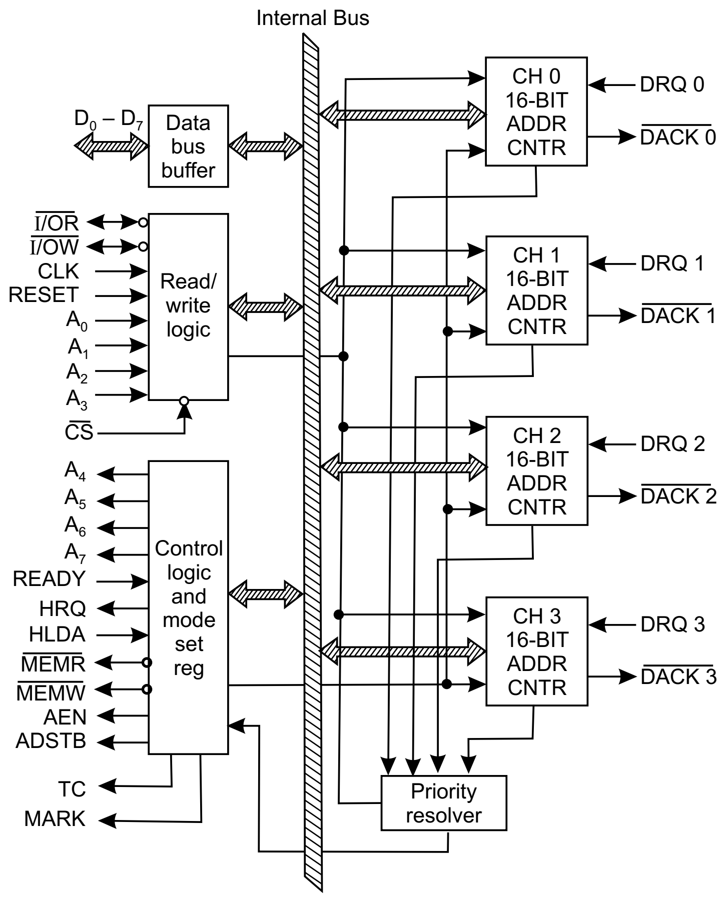


Fig. 9f.2: 8257 Functional block diagram (*Source:* Intel Corporation)

3. Describe the general features of 8257.

Ans. The general features of 8257 are as follows:

1. It is a 4-channel Direct Memory Access (DMA) interface IC which allows data transfer between memory and up to 4 I/O devices, bypassing CPU.
2. A maximum of 16 KB of data (= 2^{14}) can be transferred by this IC sequentially at a time. When a DMA request comes from a peripheral, the DMAC 8257, via its HRQ (Hold Request) pin (pin number 10, which is an output pin), requests the CPU on its HOLD pin (pin number 39 of CPU 8085). CPU then acknowledges this request via its HLDA (pin 38) pin which goes to HLDA pin (pin 7) of 8257. After this, DMAC generates the required $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{I/OR}}$, $\overline{\text{I/OW}}$ signals through its 1-4 pins.
3. Initialisation of the DMAC is done under program control for each channel. The parameters which need to be initialised for each channel are starting address, number of bytes of data to be transferred, mode of operation, etc.
4. DMAC can be operated in three modes: (*a*) DMA Read (reading from memory, writing into peripheral), (*b*) DMA Write (writing into memory, reading from peripheral), (*c*) DMA verify.
5. Priority for each of the 4 channels can be set in (*a*) fixed priority, (*b*) rotating priority.

6. A Terminal Count Register exists for each of 4 channels. The number of bytes of data to be transferred is stored in the D_{13}–D_0 positions of the 16-bit Terminal Count Register. On completion of data transfer, the Terminal Count (TC) (pin 36, an output pin) goes high.

4. How many I/O devices can 8257 access?

Ans. Up to 4 I/O devices can be accessed by 8257.

5. What is the maximum value of KB of data that 8257 can transfer?

Ans. 8257 can transfer a maximum of 16 KB (16,384 = 2^{14}) of data.

6. What are the modes of operation of 8257?

Ans. 8257 can be operated in three modes. These are:

- DMA read
- DMA write and
- DMA verify

7. Comment on priority when 8257 services the external I/Os for data transfer.

Ans. Priorities, to service the external interrupts, can be

- fixed priority.
- rotating priority.

In the Mode Set Register (it is also called control register) of 8257, D_4 bit corresponds to 'enable Rotating Priority' bit.

If this bit is not set, then CH–0 is assigned highest priority and CH–3 lowest priority—this is by default. If D_4 is set to '1', the channel that has just been serviced is allocated the lowest priority. The DMA operation always assigns highest priority to CH–0.

If more than one channels are enabled, and they all place the request for DMA transfer, consecutive DMA cycles service different channels, beginning with CH–0.

8. What are the registers available with each channel of 8257?

Ans. The registers which are available with each channel of 8257 are:

- An Address Register (16-bit)
- A Terminal Count Register (TCR) (16-bit)

9. How the 8257 is initialised?

Ans. The 8257 is initialised by the CPU

(*a*) by loading the starting address of a DMA block for an I/O device in the 16-bit address register.

(*b*) by loading $D_{13} – D_0$ bits i.e., lower 14-bits of Terminal Count Register (TCR) with the number of bytes of data to be transferred.

(*c*) by loading D_{15} and D_{14} of TCR appropriately to set the mode of operation of 8257.

(*d*) by loading the Mode Set Register appropriately.

10. When does the status of pin 36 (TC = Terminal Count) of 8257 go high?

Ans. Pin 36 of 8257, which is the TC pin (an output pin, active high type) is raised high ('1' state) when the contents of TCR of the selected channel become zero.

11. What are meant by an enabled and a disabled peripheral?

Ans. The peripherals which are granted DMA transfer are called enabled peripherals and the

ones who are denied DMA transfer are called disabled peripherals. The above is done by Mode Set Register.

12. What are the jobs that are performed by 8257 sequentially, when it receives a request from an enabled peripheral?

Ans. 8257 does the following jobs sequentially, when it receives a request from an enabled peripheral:

(*i*) Gains control of the system buses, once the HOLD signal issued by 8257, is acknowledged by 8085.

(*ii*) 8257 sends an acknowledgement signal to the peripheral which is currently having the highest priority.

(*iii*) The lower 8-bits of the memory address are put out as $A_0 - A_7$ pins. These are connected to the $A_0 - A_7$ lines of the system bus. The most significant 8-bits of the memory address are put via the data bus lines $D_0 - D_7$. These are latched by 8212 latch which places them on the system address bus $A_8 - A_{15}$.

(*iv*) I/O read/write and memory read/write signals are generated.

13. Which pin of 8257 acts in a similar fashion as the ALE (address latch enable) pin of 8085? Elaborate.

Ans. Pin 8 which is ADSTB (address strobe) is an active high output pin and functions in the same manner as the ALE pin of 8085.

At the beginning of each DMA cycle, 8257 puts the most significant byte of the DMA address register on its $D_0 - D_7$ pins. These are latched by 8212 latch using the ADSTB strobe pulse of 8257. Thus at the end of this cycle, the $D_0 - D_7$ pins of 8257 can be used for data transfer purpose.

14. Where the number of bytes of data, to be transferred by DMA mode, are stored?

Ans. The number of bytes of data to be transferred by DMA mode are stored in $D_0 - D_{13}$ bits of Terminal Count Register (TCR). They are loaded with a value which is the required number of DMA cycles minus one.

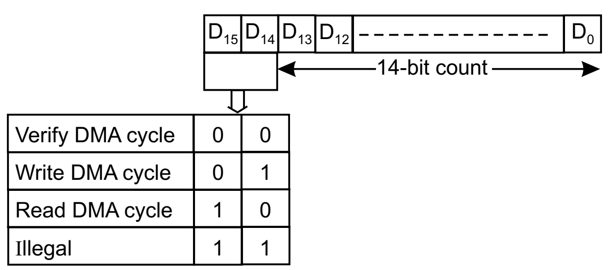


Fig. 9f.3: The terminal count register (TCR)

As shown, the two bits D_{15} and D_{14} together are loaded to set the mode of operation for that channel—like DMA write or read cycle, etc.

15. With regard to data transfer, how many classes of DMA are possible?

Ans. With regard to data transfer under DMA control, two classes are possible:

(*i*) *Sequential DMA:* In this, the DMA controller reads a data byte from memory and then writes the same into I/O or vice-versa. For each of these read or write operations, 2 to 4 CLK cycles are required.

(*ii*) *Simultaneous DMA:* It is the fastest transfer process. Here Read and Write operations are performed at the same time. Thus both $\overline{MEMR}$ and $\overline{IOW}$ (or $\overline{IOR}$ and $\overline{MEMW}$) are active at the same time. Thus a speed improvement of twice the sequential DMA class is possible in this case.

When bulk data is to be transferred, (*i*) is used while (*ii*) is used for moderate data transfer.

16. Why DMA mode of data transfer scheme is the fastest?

Ans. In normal data transfer schemes, a data coming from an I/O is first taken to ACC of the CPU and then stored in memory.

But in DMA scheme, straightway data exchange takes place between memory and I/O, bypassing the ACC of CPU. Since ACC of CPU does not take part (it is absent) in DMA mode of data transfer scheme, thus this mode is fastest.

17. What are the basic building blocks of 8257?

Ans. The basic building blocks are as follows:

DMA channels, Data bus buffer, Read/Write block ($\overline{I/OW}$, $\overline{I/OR}$, $\overline{CS}$, Reset, CLK, $A_0 - A_3$), control logic block (HRQ, HLDA, $A_4 - A_7$, $\overline{MEMW}$, $\overline{MEMR}$, Ready, ADSTB, AEN, TC, MARK) and Mode Set Register.

18. Draw the basic flowchart of a DMA mode of data transfer scheme.

Ans. The flowchart will be, as shown below:

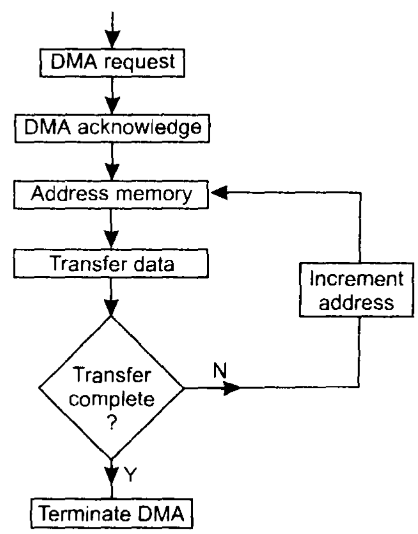


Fig. 9f.4: Flowchart of DMA

19. What is meant by a DMA cycle?

Ans. A DMA cycle indicates the transfer of a byte.

20. Through which pin a peripheral requests the 8257 for data transfer and through which pin the peripheral gets back the acknowledgement.

Ans. A peripheral requests for data transfer to 8257 via DRQ (DRQ 0 – DRQ 3) pin and it gets back the acknowledgement via $\overline{DACK}$ ($\overline{DACK\ 0}$ – $\overline{DACK\ 3}$) pin.

21. What determines the master or slave action of a DMA controller.

Ans. When the μP is in control of its buses (address bus, data bus and control bus), it acts as master and DMA controller acts as the slave. When DMA controller takes control of the buses, it becomes the master and μP becomes the slave.

22. What are the functions of Mode Set Register of 8257 DMA controller?

Ans. The functions of the Mode Set Register are as follows:

(*i*) To enable/disable a channel or channels.

(*ii*) To configure 8257 in the following four categories: Auto Load, TC Stop, Extended Write, Rotating Priority.

23. When DMA is undertaken, with whom the peripheral is synchronised?

Ans. When DMA is in progress, the peripheral is synchronised to the main memory, not the microprocessor.

24. When the DMA request line (i.e., HOLD pin of microprocessor) is sampled?

Ans. It is sampled at the end of each machine cycle and not instruction cycle.

Thus the response time for a DMA request is a maximum of one machine cycle plus one T state. For 8085 or Z–80 microprocessors the worst case scenario, i.e., maximum time is 7 (seven) T states.

25. When a non-maskable interrupt is not going to be recognised by a micro-computer system?

Ans. No interrupts—either maskable or non-maskable—will be recognised during a DMA request.

26. In how many modes DMA transfer is possible?

Ans. There are three modes. These are: (*a*) Byte or Single mode, (*b*) Burst or Demand mode, (*c*) Continuous or Block mode.

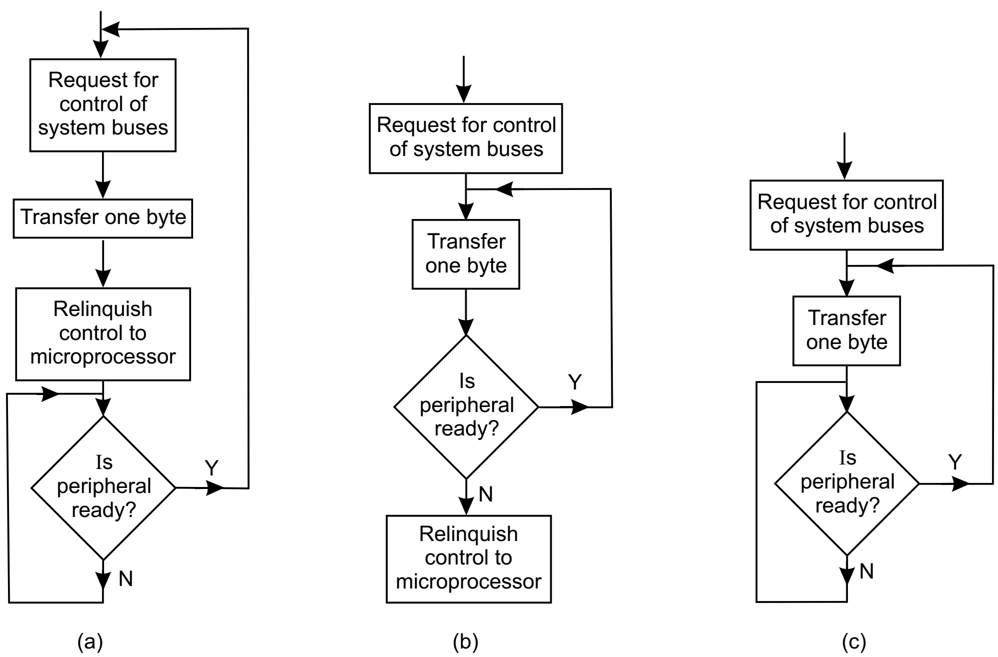


Fig. 9f.5: Flowcharts for (a) Byte or single mode (b) Burst or demand mode and (c) Continuous or block mode

In byte or single mode, after transferring one byte of data, the bus control is relinquished and handed over to microprocessor.

In burst or demand mode, data is transferred till the time the peripheral is 'ready'. After this the bus control is handed over to microprocessor.

The third method, i.e., continuous or block mode, is identical to the earlier one, but the bus control is not relinquished until the entire block of data has been transferred.

27. Which particular register is responsible for enabling (or disabling) a particular channel?

Ans. It is the shift mode set register which is responsible for enabling (or disabling) of a particular channel (CH0 to CH3). The mode set register is defined as:

Enables Auto-load		Enables Extended write			enables CH–2		enables CH–0
↑		↑			↑		↑
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	↓		↓	↓		↑	
	Enables TC stop		Enables Rotating Priority	enables CH-3		enables CH-1	

For example, if channel 1 (CH–1) is to be selected (enabled), along with TC stop option facility, then the mode set register, as per above, should have the following content:

0 1 0 0 | 0 0 1 0 = 42 H

28. How the port addresses of the registers in 8257 are done?

Ans. The port addresses of each register in 8257 are determined by the four address lines A_3–A_0. The port assignments for mode register, CH–1 DMA register and CH–1 count register are given below:

		A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		
Mode register	⇒	1	0	0	0	1	0	0	0	=	88_H
CH–1 DMA register	⇒	1	0	0	0	0	0	1	0	=	82_H
CH–1 count register	⇒	1	0	0	0	0	0	1	1	=	83_H

It is assumed that $\overline{CS}$ signal is connected to address line A_7 via an inverter and $A_6 - A_4$ are all at logic level 0.

29. Write a program to transfer AD_H bytes of data from a peripheral to memory, the memory address starting from 2459_H. Data to be inputed via CH–1.

Ans. The instructions can be sequentially written as follows:

MVIA, 42_H OUT 88_H	Enabling of CH–1 and sending the same to mode register.
MVIA, AD_H OUT 83_H	Send the low order byte count AD_H to be transferred to Terminal Count Register of CH–1
MVIA, 40_H OUT 83_H	Send high order byte count for 'Write DMA cycle' for which $D_{15} = 0, D_{14} = 1$ and $D_{13} - D_8 = 0$
MVIA, 59_H OUT 82_H	Send the low-order byte (59_H) of the memory location 2459_H to the DMA register of CH–1
MVIA, 24_H OUT 82_H	Send the high-order byte (24_H) of the memory location 2459_H to the DMA register of CH–1

30. Which particular pin of 8257 is used to convert it into 'master' mode and MPU in 'slave' mode?

Ans. AEN (Address enable) pin of 8257 is utilised to convert it into 'master' mode and at the same time translate MPU into 'slave' mode.

31. Which particular pin of 8257 is used to interface it with a slow memory?

Ans. 'READY' pin of 8257 is used for the purpose. When the memory becomes ready, it sends a high signal which is connected to the 'READY' pin of 8257.

32. When 8257 is programmed to have fixed priority, which channel will have the lowest priority?

Ans. It is the CH–3 which will have lowest priority.

33. When data is transferred from memory to an I/O, which two of the four signals $\overline{I/OR}$, $\overline{I/OW}$, $\overline{MEMR}$, $\overline{MEMW}$ of 8257 become active?

Ans. The signals which become active are $\overline{MEMR}$ and $\overline{I/OW}$. These two signals become active low.

34. What is the function of 'MARK' output pin of 8257?

Ans. When this (MARK) output pin goes high, it informs the concerned I/O device that the current DMA cycle is the 128th cycle since the previous MARK output.

35. Explain the Auto Load Option of the Mode Set Register of 8257.

Ans. Bit D_7 is the 'Auto Load Option' bit of Mode Set Register 8257. It is enabled when D_7 is set.

This 'Auto Load Option' facility is used, i.e., D_7 bit of mode set register is set (= '1') when some DMA operation is repeatedly desired—like the sending of data to a CRT monitor. This is called repetitive or chained DMA operation and utilises CH–2 and CH–3.

36. What does 'TC stop' option do when it is set?

Ans. It is the D_6 pin of mode set register. When it is set, a channel is automatically disabled when the Terminal Count output goes high. If DMA operation is to be continued or else if another operation is to begin, the channel must be enabled by a fresh Mode Set Operation in the Mode Set Register.

37. Describe the status word register of 8257.

Ans.

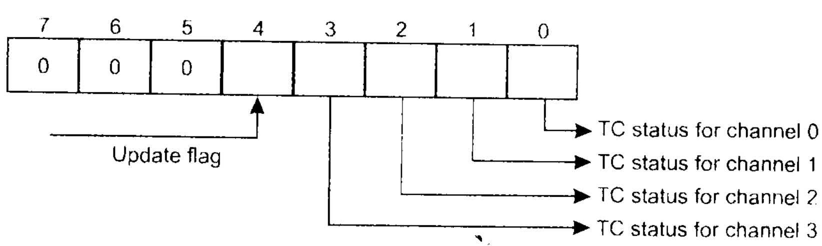


Fig. 9f.6: The status word register

The Status Word Register of 8257 is shown in Fig. 9f.6. It can be read to know the status of the terminal counts of the four channels CH0–CH3. Bit 4 corresponds to the update flag.

Any of the four bits—bit 0 to bit 3 of the status word register is set when the terminal count output corresponding to that channel goes high. When the status word is read, the TC status bits (bits 0 to 3) are cleared.

The update flag is cleared in the following cases:

- When 8257 is reset
- When 'Auto Load' option in the 'Mode Set Register' is reset.
- When it automatically goes low on the completion of update cycle.

The update flag is not affected when a status read operation is undertaken.

38. Indicate the lengths of the different registers within DMAC 8257.

Ans. 8257 has four channels. Each of these four channels has two 16-bit registers—Address Register and Terminal Count Register.

Again 8257 has two 8-bit registers—a Mode Set Register and a Status Register.

39. Describe the functions of the pins D_0 – D_7 (pins 21–23, 26–30).

Ans. The functions played by $D_0 – D_7$ bits are different for the following two cases: when 8257 is a slave, and when 8257 is a master.

In the 'slave' mode of 8257, $D_0 – D_7$ pins act as input pins. Eight bits of data at a time for the Address Register or Terminal Count Register (both 16-bits), (for a particular channel) or eight bits of data for the 8-bit Mode Set Register are received through these pins. The CPU can also read eight bits of data at a time from the Address or Terminal Count Registers or from the Status Register.

In the 'master' mode of 8257, the DMAC puts out the eight most significant bits of the DMA Address Register (for a particular channel), at the beginning of each DMA cycle, through $D_0 – D_7$. These are latched by 8212 latch and these latched values are put out on $A_8 – A_{15}$ of the system address bus. Once this operation is over, $D_0 – D_7$ pins are released so that through these pins memory data transfer can be executed for the remainder of the DMA cycle.

40. Discuss the functions of the pins A_0 – A_3.

Ans. Two different functions are played by these pins for the two cases when 8257 acts as the master or else as a slave.

In the master mode of 8257, these four pins $A_0 – A_3$ act as output pins. 8257 puts out the four least significant bits of the DMA Address Register on these four pins.

In the slave mode of 8257, while accessing the Mode Set Register or Status Word Register, the pins A_2 A_1 A_0 must all be '0's, while A_3 = 1. Again while mode set operation is done (This is a write only operation), the status of $\overline{I/OW}$ and $\overline{I/OR}$ pins would be 0, 1 while for status word (this is a read only operation), the status of the above two pins would be 1, 0 respectively.

41. Explain, in detail how the Address Registers and Terminal Count Registers for each of CH0–CH3 are selected as also the Mode Set Register and Status Word Register.

Ans. The four Address Registers and four Terminal Count Registers of Channels CH0–CH3 can be accessed only if A_3 = 0. For Mode Set Register and Word Register accessing, A_3= 1.

For individual channel selection, bits A_2 A_1 are used. With 00, 01, 10, 11 values for A_2 A_1 select channels CH0, CH1, CH2 and CH3 respectively, while the status of bit A_0 distinguishes between channel and Terminal Count Register. If A_0 = 0, any of the channels would be selected and if A_0 = 1 then Terminal Count Register would be selected.

The channel registers (Address Register or Terminal Count Register) of each channel are 16-bits each. Thus two operations must be performed on a channel register—one for the lower byte and the other for the upper byte to access it fully—be it reading or writing operation. This distinction between the two halves of a channel are done by a special internal First/Last F/F (F/L F/F). This F/F toggles at the completion of each READ/WRITE operation. This F/L F/F assumes a '0' state for LSB and '1' for MSB accessing of a channel register.

A channel register, while being accessed, must be accessed fully—i.e., both LSB and MSB of the channel should be accessed. Therefore, before programming of a channel being initiated, the system interrupts must be disabled, otherwise an interrupt occurring after the first half of channel accessing will prevent the second half of the same channel from being accessed.

The F/L F/F is reset when 8257 gets a Reset input or whenever the Mode Set Register is programmed.

The following table shows in details how both the registers in each channel (CH0–CH3), as also the Mode Set Register and Status Word can be selected, so that they can be programmed accordingly.

Table 9f.1: 8257 Registers selection (*Source:* Intel Corporation)

Register	**Byte**	**Address inputs**				**F/L**	**Bidirectional bus**							
		A_3	A_2	A_1	A_0		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CH 0 DMA Address	LSB	0	0	0	0	0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	MSB	0	0	0	0	1	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8
CH 0 Terminal Count	LSB	0	0	0	1	0	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
	MSB	0	0	0	1	1	Rd	Wr	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8
CH 1 DMA Address	LSB	0	0	1	0	0	Same as channel 0							
	MSB	0	0	1	0	1								
CH 1 Terminal count	LSB	0	0	1	1	0								
	MSB	0	0	1	1	1								
CH 2 DMA Address	LSB	0	1	0	0	0	Same as channel 0							
	MSB	0	1	0	0	1								
CH 2 Terminal count	LSB	0	1	0	1	0								
	MSB	0	1	0	1	1								
CH 3 DMA Address	LSB	0	1	1	0	0	Same as channel 0							
	MSB	0	1	1	0	1								
CH 3 Terminal count	LSB	0	1	1	1	0								
	MSB	0	1	1	1	1								
Mode set (Program only)	–	1	0	0	0	0	AL	TCS	EW	RP	EN3	EN2	EN1	EN0
Status (Read only)	–	1	0	0	0	0	0	0	0	UP	TC3	TC2	TC1	TC0

A0–A15: DMA starting address, C0–C13: Terminal Count Value (N–1), Rd and Wr: DMA verify (00), Write (01) or Read (10) Cycle selection, AL: AUTO Load, TCS: TC STOP, EW: Extended write, RP: Rotating priority, EN3–EN0: Channel enable mask, UP: Update Flag, TC3–TC0: Terminal count status bits.

42. What happens to Mode Set Register when a 'resetting' of the system is done?

Ans. When the system receives a 'reset', the mode set register is automatically cleared. It disables all the four DMA channels and inhibiting all options.

43. Compare data transfer rate of 8237 DMA and a 2 MHz 8080.

Ans. For the 8237 DMA, the data transfer rate between memory and I/O port is of the order of 1.6 MB/second while it is around 33,000 bytes/second using polling.

9g

Programmable Interval Timer 8253

1. Draw the pin diagram of 8253.

Ans. The pin diagram of 8253 is shown below:

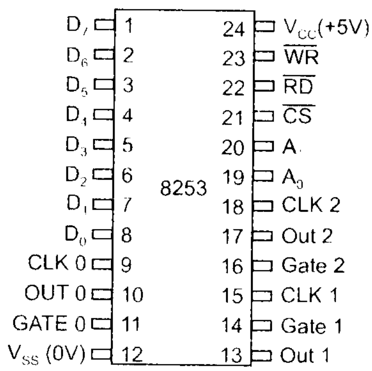


Fig. 9g.1: 8253 Pin diagram (*Source:* Intel Corporation)

2. Draw the functional block diagram of 8253.

Ans. The functional block diagram of 8253 is shown below:

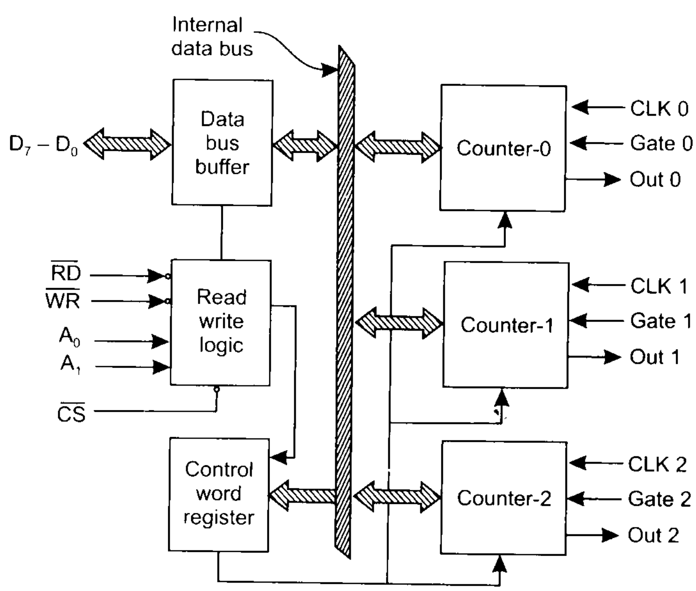


Fig. 9g.2: 8253 Functional block diagram (*Source:* Intel Corporation)

The figure shows that there are six blocks interconnected by an internal data bus. The six blocks include three counters (counter 0, 1 and 2), a data bus buffer, a Read/Write logic and a control word register.

3. How many counters are there in 8253 and how many Modes are there?

Ans. 8253 has three counters—Counter 0, Counter 1 and Counter 2. It operates in 6 different Modes—from Mode 0 to Mode 5.

Each of the three counters are 16-bit, presettable down counters. The counters can be operated in any of the six modes by proper programming.

4. Indicate the application areas of 8253.

Ans. 8253 can be used to generate accurate time delay, events counter, real-time clock, digital one-shot, a square wave generator or a complex wave generator, all under software control.

5. What is the maximum frequency of the waveform obtainable from 8253 Timer?

Ans. The maximum frequency of the waveforms obtainable from Timer 8253 is 2 MHz.

6. Indicate the types of outputs obtained under different modes.

Ans. The different types of outputs obtained from Mode 0 to Mode 5 are as under:

(*i*) An interrupt is obtained on the Terminal Count in Mode 0.
(*ii*) A negative pulse of controllable width is obtained in Mode 1.
(*iii*) A symmetric square wave of controllable frequency is obtained in Mode 2.
(*iv*) A symmetric square wave is obtained in Mode 3.
(*v*) A negative pulse of one clock period is obtained after a software controlled delay in Mode 4.
(*vi*) A delayed negative pulse of one clock period is obtained following a positive going trigger input in Mode 5.

7. What are the functions of the internal data bus buffer?

Ans. This internal data bus is interfaced with the system data bus. The functions of this bus are as follows:

- Loads the count registers.
- Programming the modes of 8253.
- Reads the count values.

8. What are the signals associated with Read/Write Logic block?

Ans. The five control signals associated with this block are: A_0, A_1, $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$. 8253 can be enabled/disabled by $\overline{CS}$ signal.

9. What are the signals associated with each counter.

Ans. There are three counters in 8253. Each of these three counters has two input signals—Clock (CLK) and GATE and one output signal—OUT. The purpose of the GATE signal is to enable/disable a particular counter.

10. How the Control Word Register is selected?

Ans. The control word register is selected only if A_1 A_0 = 11. Also the status of $\overline{CS}$, $\overline{RD}$, $\overline{WR}$ signals should be 0, 1 and 0 respectively.

11. Write down the Control Word format as also the mode definitions.

Ans. The details of the control word format, as also the mode definitions are detailed below:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Definition of Control:

SC-Select Counter:

SC1	SC0	
0	0	Select counter 0
0	1	Select counter 1
1	0	Select counter 2
1	1	Illegal

M-Mode

0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

RL-Read/Load:

RL1	RL0	
0	0	Counter latching operation (see read/write procedure section)
1	0	Read/Load most significant byte only
0	1	Read/Load Least significant byte only
1	1	Read/Load Least significant byte first, then most significant byte

BCD:

0	Binary counter 16-bits
1	Binary coded decimal (BCD) counter (4 decades)

Fig. 9g.3: Control word format and mode definitions
(*Source:* Intel Corporation)

The information stored in the control word gives the following details:

- Bits D_7 and D_6 (SC_1 and SC_0) select a counter.
- Bits D_5 and D_4 (RL1 and RL0) determine whether it is a Read or Load Count operation and also whether it is Least Significant Byte or Most Significant Byte of the count that is involved.
- Bits D_3, D_2, D_1 (M2, M1, M0) determine the operating mode (i.e., Mode 0 to Mode 5)
- Bit D_0 (BCD) determines the counting sequence in binary or BCD format.

12. How to ensure that a counter is loaded?

Ans. For the above to be ensured, it is necessary that:

- The count value be written (a single byte or double byte—which depends on the mode selection by RL1 and RL0 bits).
- The above is then to be followed by a rising and a falling edge of the clock. Data, read before the falling edge of the clock, is an invalid one.

13. How writing operation is done in a counter (i.e., counter 0 or 1 or 2)?

Ans. First the control word register is selected.
Secondly, the control word is to be appropriately chosen/written by

(*a*) selecting the counter in which writing is to be done ($D_7 - D_6$ bits).
(*b*) filling in $D_5 - D_4$ bits correctly (which takes care of 1 or 2 byte count).
(*c*) filling in $D_3 - D_1$ bits, which corresponds to Mode selection.
(*d*) filling in D_0 bit—its content reflects the count down to be done in binary/BCD.

14. What will be the Control word register address if $\overline{CS}$ of 8253 is connected to A_7 via an inverter.

Ans. Since $\overline{CS}$ pin of 8253 is connected to A_7 via an inverter, then A_7 will have to be 1, i.e., $A_7 = 1$. Again A_1 and A_0 will both have to be 1 to ensure writing in the control register. Assuming A_6 to A_2 to be 0, then the port address of the control word register will be

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		
1	0	0	0	0	0	1	1	=	83H

15. Write the value of the control word if in a specific case counter 0 is to be selected in Mode 1. The counter should have a 16-bit count and that it should count down in binary. Lastly load the control word register with the control word value so formed. Assume address of control word register = 83_H.

Ans. Control word for load operation is as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
0	0	1	1	0	0	1	0	=	32_H

$D_7 D_6$ → Selecting counter 0
$D_5 D_4$ → Loading 16-bit count
$D_3 D_2 D_1$ → Mode 1 selection
D_0 → Count in binary

The program for loading the control word value 32_H in the control register address 83_H is as follows:

MVI A, 32_H = Control word is loaded in accumulator

OUT 83_H = The control word (= 32_H) is written into the control word register, having address 83_H.

16. Show in a tabular form the conditions of the different Modes corresponding to the different status of the Gate signal.

Ans. This is shown below in a tabular form:

Table 9g.1: Gate pin operations (*Source*: Intel Corporation)

Signal status / *Modes*	**Low or Going low**	**Rising**	**High**
0	Disables counting	——	Enables counting
1	——	1. Initiates counting 2. Resets output After next clock	——
2	1. Disables counting 2. Sets output Immediately high	1. Reloads counter 2. Initiates counting	Enables counting
3	1. Disables counting 2. Sets output Immediately high	Initiates counting	Enables counting
4	Disables counting	——	Enables counting
5	——	Initiates counting	——

17. Discuss the two methods of reading the value of the count in a counter while count is in progress.

Ans. There are two methods of doing the same

(a) Reading by halting/stopping the count

(*b*) Reading while counting is 'ON' (i.e., counting is in progress)

(*a*) Reading by halting: The procedure is as follows:

(*i*) The counter must be identified with appropriate A_1 A_0 status.

(*ii*) The counter is then halted by either disabling the Gate pin or inhibiting the CLK input.

(*iii*) Then I/O read operation is done—the first I/O Read gets the LSB and the second I/O Read gets the MSB.

(*b*) Reading while counting is in progress: For this to be effective, the mode register is loaded with a code that would load the present count (in a counter) to be latched on to a storage register.

The Mode Register Format for Latching Count is as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	0	0	X	X	X	X

$D_5 - D_4$ are loaded with 0 each for latching the present counter content.

Then the Read command is invoked to the selected counter which gives the content of the latched register—the counter must be programmed for two bytes and must be read prior to any Write Command to the concerned counter.

9h

8254: Programmable Interval Timer

1. Draw the pin diagram and functional block diagram of 8254.

Ans. The pin diagram and functional block diagram of 8254 are shown below:

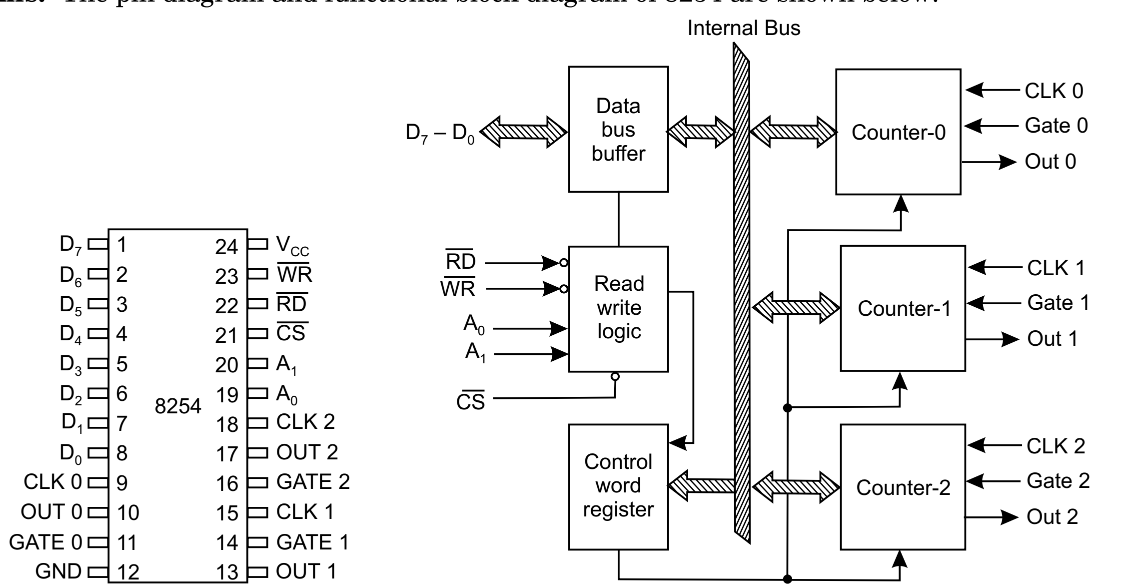


Fig. 9h.1: Block diagram and pin descriptions for the 8254 programmable interval timer. Three separate timer/counters are provided. (*Courtesy:* Intel Corporation)

2. How many counters are there in 8254?

Ans. There are three 16-bit counter registers, each of which can be programmed as a timer or an event counter. The counters are named as COUNTER 0, COUNTER 1 and COUNTER 2.

3. How programming is done in 8254?

Ans. The programming is done by writing a control word in the Control Word Register.

4. In how many modes can 8254 operate?

Ans. 8254 can operate in six possible operating Modes—Mode 0 to Mode 5.

5. What kind of outputs are available from the different operating modes.

Ans. From the six different operating modes, the outputs which are available are: event counter, one-shot, square-wave generator, divide-by-N counter, hardware triggered strobe and software triggered strobe.

6. What happens when the Terminal Count (TC) of a counter is reached?

Ans. The counters operate in count down mode. When a counter counts down to zero, it is called 'Terminal Count'. When TC for a particular counter occurs, the following may occur:

- an interrupt can be requested
- a one-shot pulse can be terminated
- a strobe pulse can be generated
- the logic level of a square wave can be toggled.

7. What are the differences between timers 8253 and 8254?

Ans. Timer 8254 is actually a "Superset" of timer 8253 and is pin compatible with the latter. There are two differences between the two, which are

- 8254 has read back mode facility—which means that the status of a particular mode can be read after programming. This facility is not available with 8253.
- The maximum clock frequency of 8254 is 10 MHz, whereas that of 8253 is 2 MHz.

8. In how many forms can the control word register be used?

Ans. The control word can be used in three formats—Standard mode, counter latch mode and Read back mode.

9. How one of the Six Modes (Mode 0 to Mode 5) is selected?

Ans. The particular mode is selected by the standard form of the control word in the control word register.

10. Draw the control word register format and discuss.

Ans. The control word of the control word register is shown Fig. 9h.2.

The control word byte is divided into four parts $D_7 - D_6$, $D_5 - D_4$, $D_3 - D_2 - D_1$ and D_0. The different combinations of these eight bits give rise to 'standard' mode, 'Counter latch' mode or 'Read back' mode and are self evident from the figure.

11. Discuss the six different modes in which 8254 can operate.

Ans. The six different modes are Mode 0 to Mode 5. These are discussed below:

Mode 0: This is an 'event counter', when GATE = 1, the counter will start decrementing from its stored value on the falling edge of the second pulse of CLK input. OUT will go high when 'terminal count' is reached. This OUT signal can be utilised as an interrupt input to the microprocessor.

Mode 1: It is 'hardware triggered one shot', when the rising edge of the GATE pulse is received, OUT goes low and remains low till TC (terminal count) is reached. Then OUT goes high. This low one shot duration is equal to the stored value in the counter multiplied by the CLK period.

Mode 2: It is a 'Divide-by-N' counter, when GATE = 1, OUT goes low for one period of the CLK input after the stored count is decremented to 1. The initial (or stored) count will then automatically get reloaded and the cycle repeats.

Mode 3: It is a 'square wave generator'. It is identical to Mode 2, but the duty cycle (= on time/period) here is 50%. In case, the initial count is an odd number, OUT then will be high for one more clock cycle then it is low.

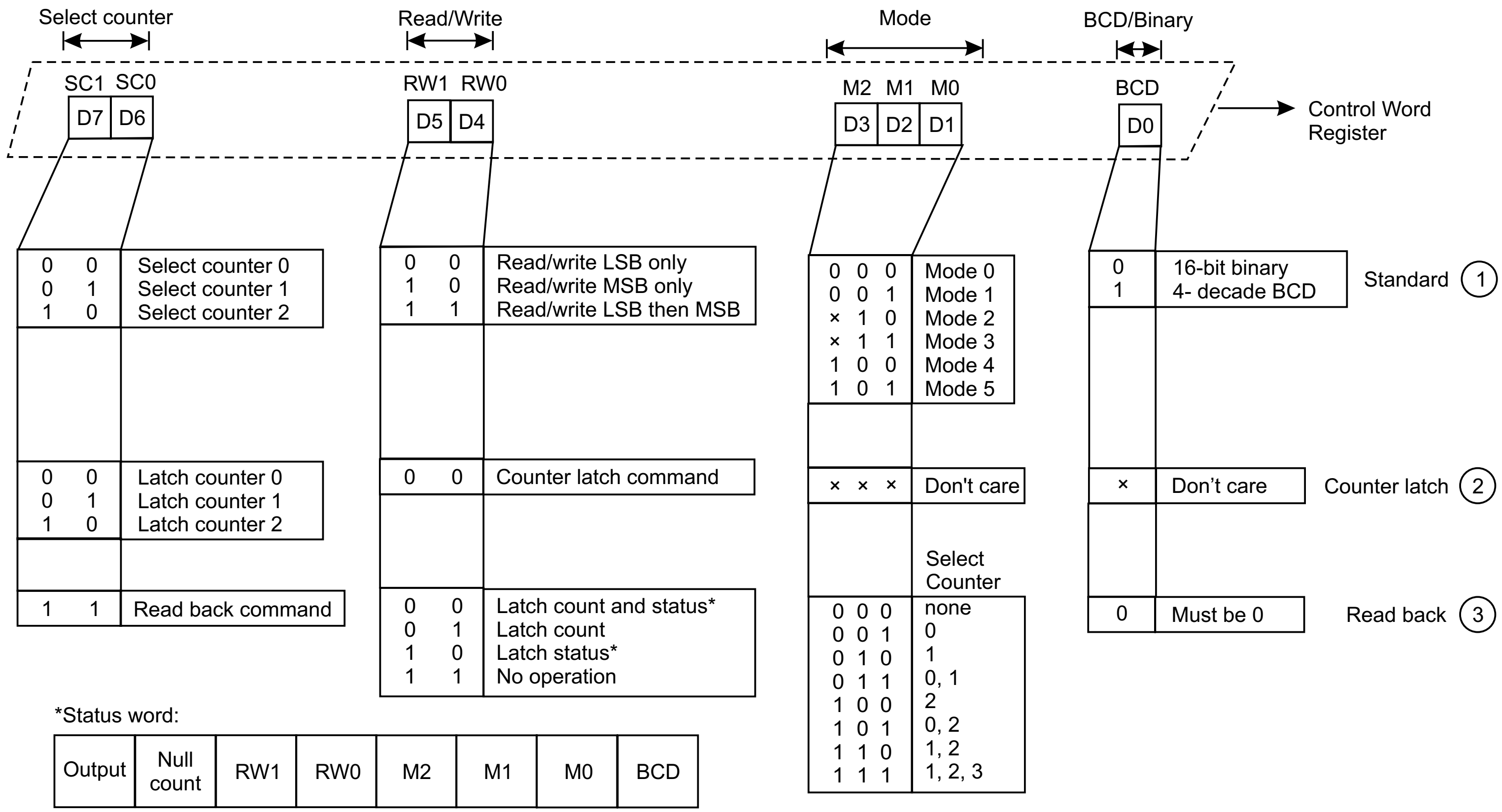


Fig. 9h.2: 8254 Control word. The standard form is used to specify the operating mode. The Counter Latch and Read Back Commands are used when the present count or status is to be read

N.B.: (1), (2) and (3) are the three types of commands possible by the Control Word Register.

Mode 4: It is a 'software triggered mode'. If GATE = 1, OUT goes low for one period of the CLK input N clock cycles after, where N is the initial (or stored) number in the counter. For second strobing to be done, N must be reloaded in the counter.

Mode 5: It is a 'Hardware triggered mode'. On the appearance of rising edge of the GATE input, the stored or initial count starts decrementing to 0. When TC occurs, OUT goes low for one period of the CLK input.

12. How the three counters and the control word are selected?

Ans. The combination of A_1 A_0 select the above and shown below:

A1	A0	
0	0	Write into Counter 0
0	1	Write into Counter 1
1	0	Write into Counter 2
1	1	Write Counter word

13. Write down the port addresses of the three counters and the control word register. The $\overline{CS}$ signal is derived on the basis of $A_7 - A_4$ = 1111 and $A_3 - A_2$ = 00.

Ans. The port addresses of the counters and the control word register are determined as follows:

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0				Port address of
1	1	1	1	0	0	0	0	=	F0	⇒	Counter 0
1	1	1	1	0	0	0	1	=	F1	⇒	Counter 1
1	1	1	1	0	0	1	0	=	F2	⇒	Counter 2
1	1	1	1	0	0	1	1	=	F3	⇒	Control Word Register.

14. Write down the Control Word so that Counter 1 operates in Mode 0 in binary sequence.

Ans. By having a look at the control word register, the control word will be as follows:

0 1   1 1   0 0 0   0 = 0111 0000 = 70

While writing the above control word, it is assumed that Counter 1 is to be loaded with a 2-byte count.

9i

USART 8251 (Universal Synchronous/ Asynchronous Receiver Transmitter)

1. Draw the pin diagram of USART 8251.

Ans. The pin diagram of 8251 is as shown below:

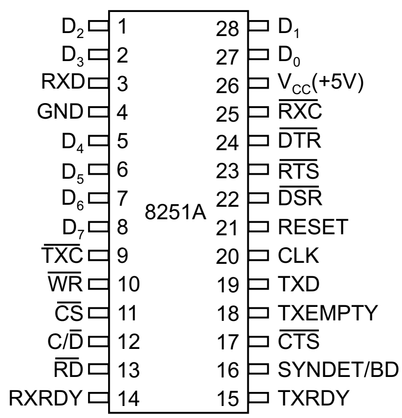


Fig. 9i.1: 8251 pin diagram (*Source:* Intel Corporation)

2. Draw the functional block diagram of 8251.

Ans. The functional block diagram of 8251 is shown below:

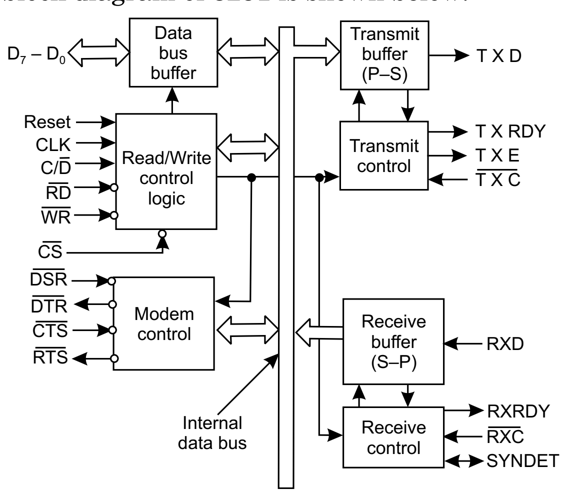


Fig. 9i.2: Functional block diagram of 8251

3. How many different sections does 8251 have?

Ans. 8251 has five sections: Read/Write control logic, data bus buffer, modem control, transmitter (including its control) and receiver (including its control).

4. How the 8251 is programmed?

Ans. 8251 is programmed by a 16-bit Control Word Register. This 16-bit register is divided into two bytes—the first byte corresponds to Mode Instruction Format while the second byte corresponds to Command Instruction Format. The content of control word register determines synchronous or asynchronous operation, Baud rate, number of bits per character, number of stop bits, nature of parity, etc.

5. What is the function of the Status Word Register of 8251?

Ans. The function of the status word register is to check or examine the preparedness of 8251 with regard to transmission or reception of data.

6. Describe the Read/Write Control logic and registers.

Ans. It contains three buffer registers:

- data buffer register
- control register
- status register.

The six input signals are: $\overline{CS}$, $C/\overline{D}$, $\overline{WR}$, $\overline{RD}$, RESET and CLK.

The particular 8251 is selected on $\overline{CS}$ signal going low. This pin is usually connected to a decoded address bus. $C/\overline{D}$ stands for control/data pin. When this pin is high, either the control register or status register is selected and when low, data bus buffer is selected. The control register and the status register are distinguished by $\overline{WR}$ and $\overline{RD}$ signals, respectively.

The figure below shows how the three registers: data buffer register, control register and status register are accessed by making respectively.

$$C/\overline{D} = 0, \quad \overline{RD} = 0 \quad \text{or else } \overline{WR} = 0$$
$$C/\overline{D} = 1, \quad \overline{WR} = 0$$
$$C/\overline{D} = 1, \quad \overline{RD} = 0$$

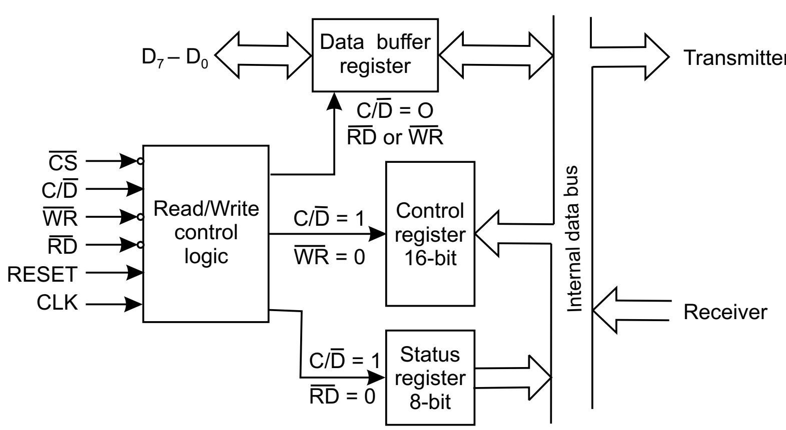


Fig. 9i.3: Accessing the data buffer register, control register and status register

The following table shows the status of the control signals $\overline{CS}$, C/$\overline{D}$, $\overline{RD}$, $\overline{WR}$ for accessing the different registers.

Table 9i.1: Summary of Control Signals for the 8251A

$\overline{CS}$	$C/\overline{D}$	$\overline{RD}$	$\overline{WR}$	*Function*
0	1	1	0	MPU writes instructions in the control register
0	1	0	1	MPU reads status from the status register
0	0	1	0	MPU outputs data to the Data Buffer
0	0	0	1	MPU accepts data from the Data Buffer
1	X	X	X	USART is not selected

7. Explain the operation of the transmitter section of 8251.

Ans.

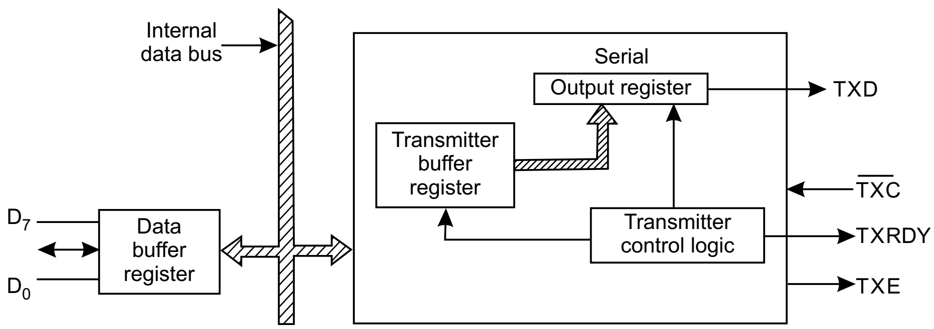


Fig. 9i.4: Transmitter section

The transmitter section consists of three blocks—transmitter buffer register, output register and the transmitter control logic block. The CPU deposits (when TXRDY = 1, meaning that the transmitter buffer register is empty) data into the transmitter buffer register, which is subsequently put into the output register (when TXE = 1, meaning that the output buffer is empty). In the output register, the eight bit data is converted into serial form and comes out via TXD pin. The serial data bits are preceded by START bit and succeeded by STOP bit, which are known as framing bits. But this happens only if transmitter is enabled and the $\overline{CTS}$ is low. $\overline{TXC}$ signal is the transmitter clock signal which controls the bit rate on the TXD line (output line). This clock frequency can be 1, 16 or 64 times the baud.

8. Explain the operation of the receiver section of 8251.

Ans.

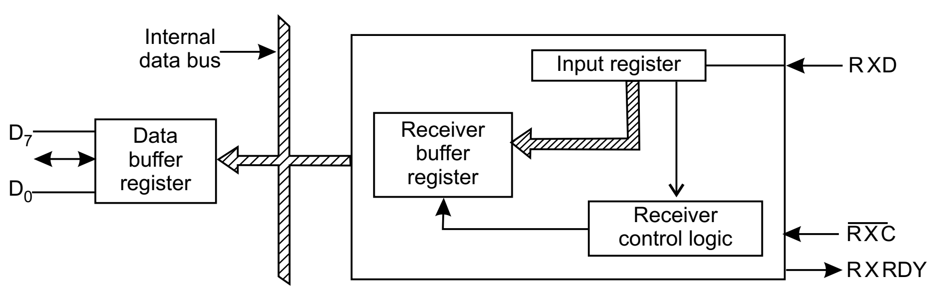


Fig. 9i.5: Receiver section

The receiver section consists of three blocks — receiver buffer register, input register and the receiver control logic block. Serial data from outside world is delivered to the input register via RXD line, which is subsequently put into parallel form and placed in the receiver buffer register. When this register is full, the RXRDY (receiver ready) line becomes high. This line is then used either to interrupt the MPU or to indicate its own status. MPU then accepts the data from the register.

$\overline{\text{RXC}}$ line stands for receiver clock. This clock signal controls the rate at which bits are received by the input register. The clock can be set to 1, 16 or 64 times the baud in the asynchronous mode.

9. How does the CPU know that the transmitter buffer is empty?

Ans. The CPU knows about the same by

- examining the TXRDY line of the transmitter.
- examining D_0 bit of the status word.

10. When the TxD line goes into the MARKING (HIGH) state?

Ans. It becomes high in the following cases:

- a RESET is received by 8251.
- transmitter is empty.
- transmitter is not enabled.
- $\overline{\text{CTS}}$ is off.

11. How the transmitter is enabled?

Ans. The transmitter is enabled by setting bit D_0 of the command instruction word.

12. What is the status of the start bit on the RXD line for 8251 (in asynchronous mode only) and how does it differentiate between a valid start pulse and transient pulse?

Ans. For any data to be received by the receiver, it first checks for a valid start bit which is zero. 8251 has an inbuilt false start bit detection circuit which can differentiate between an actual start bit pulse and a transient pulse.

13. What happens when a parity error or a framing error occurs in the received data bits (in asynchronous mode only)?

Ans. The Parity error and Framing error status bits in the status word are set if there is a parity error or if the stop bit is absent at the end of the received bits respectively.

14. When the RXRdy line goes high in asynchronous and synchronous mode of operation?

Ans. In the asynchronous mode, RXRdy line goes high

- if the receiver is in the enabled condition (this is made so by setting D_2 bit of the Command Instruction Word).
- and after the receiver has detected a valid start bit, assembled the character bits and transferred the character to the receiver buffer register.

Whereas in the synchronous mode RXRdy line goes high

- if the receiver is enabled.
- a character is assembled and transferred to the receiver buffer register.

15. What is overrun error?

Ans. D_4 bit of the status word stands for 'over run error'.

If the CPU cannot read the data from the receiver buffer register (this happens if the CPU fails to respond to RXRdy line), then on receipt of the next character, the previous data will be written over and the earlier character will be lost. When such is the case D_4 bit of the status word is set.

16. Discuss the SYNDET/BD pin.

Ans. This is pin 16 of 8251 and stands for sync. detect (SYNDET)/Break Detect (BRKDET).

This pin is used for detection of SYNC characters in synchronous mode and Break characters in asynchronous mode.

Synchronous mode: In the synchronous case this pin (SYNDET) can be used as either input or output pin with the help of control word. When the system is RESET, the status of this pin is low in the output mode. When 8251 is programmed to receive two synchronous characters, this output pin goes high at the mid point of the last bit of the second synchronous character. The status of this pin can also be known by reading the status word, but gets resetted on STATUS READ.

In the input mode—called the external synchronous detect mode—a rising edge on this pin causes 8251 to start collecting data characters on the rising edge of the next $\overline{\text{RXC}}$. This input signal can be removed once synchronisation is achieved. When external synchronisation is done, the internal synchronisation is disabled.

BRKDET: In this mode, this pin acts as an output pin to detect break characters. If RXD remains low for two consecutive stop bit sequences, this pin (BRKDET) goes high. Here also provision is there to read the status of this pin by STATUS READ operation.

This pin is reset

- on a Master Chip Reset
- when RXD becomes on 1.

17. What purpose does 8251 serve—DTE or DCE in a communication interface environment?

Ans. 8251 acts as a DTE (Data Terminal Equipment) in such a case.

18. Why modems (modulators-demodulators) are used in case of digital transmission of data?

Ans. The term 'modem' stands for modulator—demodulator. In a communication environment, two modems are used—one at the transmitting end side and one at the receiving end side. Modems are generally called DCE (Data Communication Equipment).

High frequency digital signals require a very wide transmission channel bandwidth which makes the system very costly. However, existing telephone line facilities (which carry analog signals in the range of 40 Hz to 4 KHz) can be used to transmit such high frequency digital signals. A modem converts a digital signal into audio tone frequencies (at the transmitting end side) and reconverts this audio frequencies into h.f. digital signals (at the receiving end side)—and it utilises Frequency Shift Keying (FSK) for this purpose. Thus a modem converts a logical '1' to 1200 Hz and '0' to 2200 Hz audio frequency. These signals are then transmitted over a telephone line over a 'carrier'. The inverse operation is done at the receiving end side.

19. Draw the block diagram of a DTE-DCE interface in a communication environment.

Ans. The block diagram is shown below:

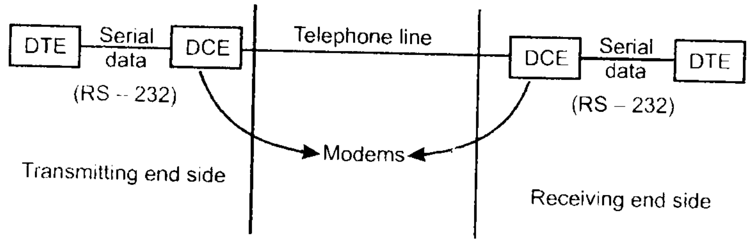


Fig. 9i.6: DTE-DCE interface

Digital data is delivered at the DTE (may be 8251) in parallel form, which is then converted into serial form and sent to DCE via RS-232 cable. The DCE (a modem) output is an audio signal carried through a telephone line.

At the receiving end side, the opposite process is carried out to retrieve the original data.

20. Draw the 8251 data loading sequence and explain the same.

Ans. The data loading sequence of 8251 is shown below:

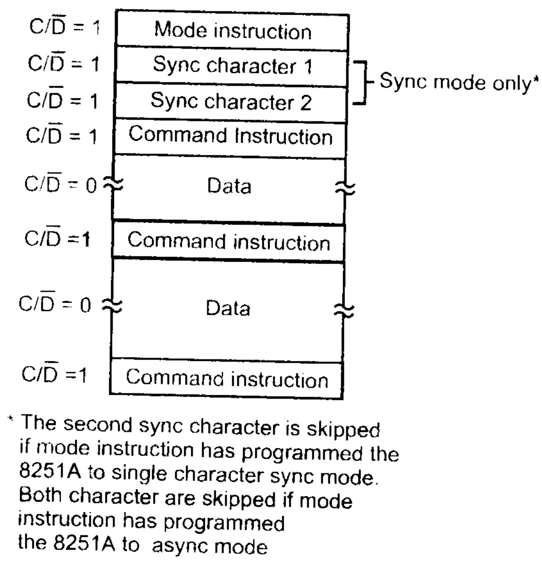


Fig. 9i.7: 8251 Data-loading sequence (*Source:* Intel Corporation)

The mode control is specified first, which indicates the general operating conditions. If the mode word indicates that it is a synchronous operation, then the synchronous character(s) is/are loaded. This is followed by loading the command instruction format. In all these, $C/\overline{D} = 1$. After this, $C/\overline{D}$ is made 0 when data is either transmitted/received. It is followed by command instruction and data in that order which is repeated all over again.

A command word with $D_6 = 1$ returns 8251 to mode instruction format.

21. Discuss the mode instruction format for asynchronous transmission/reception case.

Ans. The mode instruction format for asynchronous operation (transmission/reception) is shown below:

Bits D_0 D_1 cannot both be low for asynchronous communication. These two bits determine the baud rate factor. Bits D_2 D_3 determine the character length (which may be 5 to 8 bits in length)—depending on the content of these two bits. Bit D_4 stands for 'Parity Enable' (PEN) and is enabled if $D_4 = 1$ and otherwise if $D_4 = 0$. D_5 bit stands for 'Even Parity' (EP). Parity is even if $D_5 = 1$ and odd if $D_5 = 0$. Bits D_6 and D_7 determine the number of stop bits. There can be 1, $1\frac{1}{2}$, 2 number of stop bits.

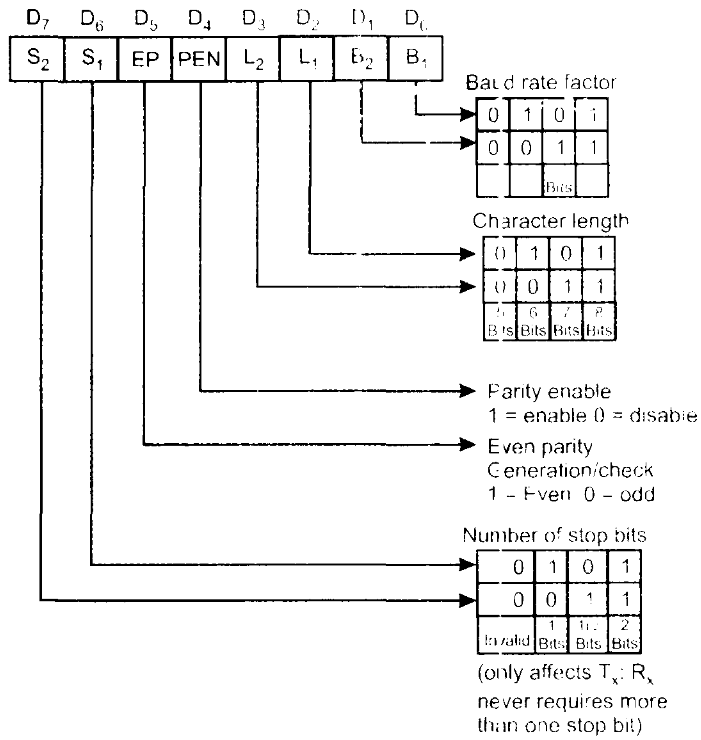


Fig. 9i.8: Mode instruction format: Asynchronous operation
(*Source:* Intel Corporation)

22. Draw the general transmission/receive format for asynchronous communication.

Ans. The general transmission/receive format for asynchronous communication is shown below:

The transmission format consists of start bit, data character, parity bit, stop bit(s)—in that order.

8251 starts sending data on the TXD (transmit data pin) pin with a start bit which is a 1 to 0 transmission. Then the data bits are transmitted, followed by stop bit(s). The data bits start with LSB of the serial output register. All these bits (start bit, data bits, stop bit(s) are shifted out on the falling edge of $\overline{TXC}$ (transmitter clock). In case when no data is transmitted, TXD output remains high. But if a 'break' is programmed, TXD line will go low.

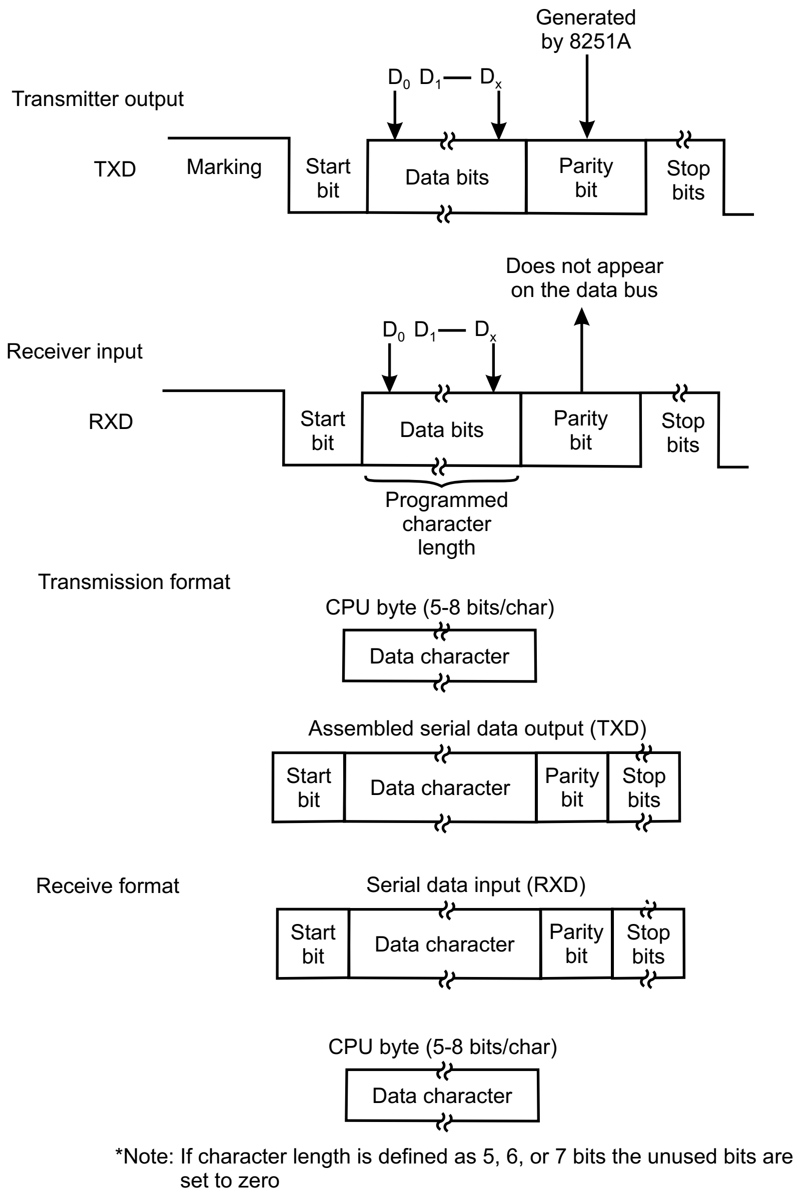


Fig. 9i.9: Asynchronous transmission and reception
(*Source:* Intel Corporation)

The receive format is identical to transmit format. Data reception starts with RXD (receive data pin) line going low—it indicates the arrival of start bit. This 1 to 0 transition on the RXD line triggers the 'False Start Bit Detection Circuit'. This circuit then samples the RXD line half-a-bit time later to ensure the presence of a genuine start bit. If this sampling results in a low on RXD line, it indicates a valid start bit. The bit counter is started on the second sampling—hence each subsequent data bit is

sampled at the middle of each bit period. This is called 'mid bit sampling'. The bit counter thus samples the data bits, parity bit and lastly the stop bit. The receiver needs only one stop bit—but the transmitter is affected by the number of stop bits. For any error during receiving of data with regard to Parity, Framing or Overrun—the corresponding flags in the status word are set.

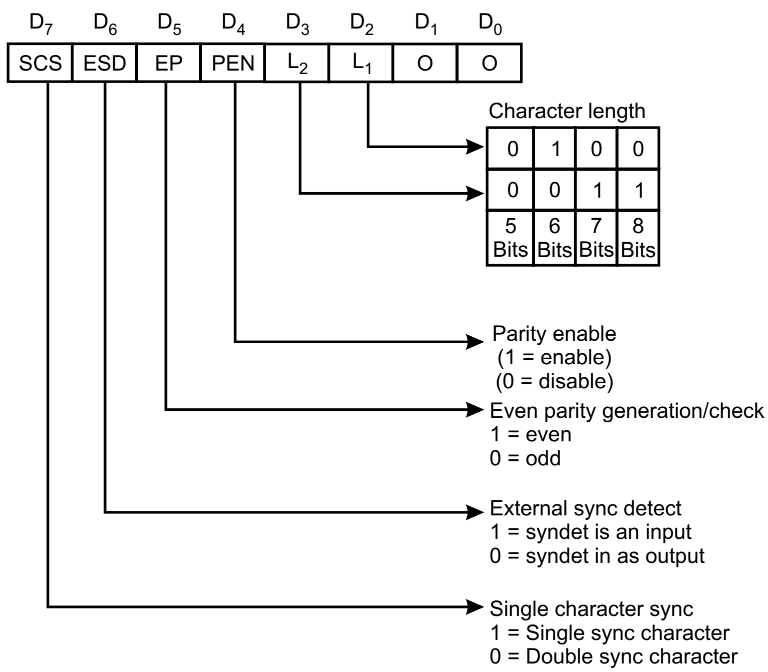
23. Why the 'false start bit detection circuit' is there in asynchronous reception case?

Ans. This is done to avoid any possibility of a false start bit detection due to a transient noise pulse.

24. Discuss the mode instruction format for synchronous transmission/reception case.

Ans. The mode instruction format for synchronous operation (transmission/reception) is shown below:

Bits D_0 D_1 both will have to be low for synchronous transmission/reception of data. Bits D_2 D_3 indicates the character length. Bits D_4 and D_5 stand for PEN and EP respectively—exactly same as in the case of asynchronous case. Bit D_6 stands for ESD (External Synchronous Detect). D_6 = 1 stands for input and D_6 = 0 stands for output. Bit D_7 stands for SCS (Single Character Sync.) with D_7 = 1 indicating a single synchronous character and D_7 = 0 indicating double synchronous characters.



Note: In External sync mode, programming double character sync will affect only the TX.

Fig. 9i.10: Mode instruction format: Synchronous operation (*Source:* Intel Corporation)

25. Draw the general transmission/receive format for synchronous communication.

Ans. The general transmission/receive format for synchronous communication is shown below.

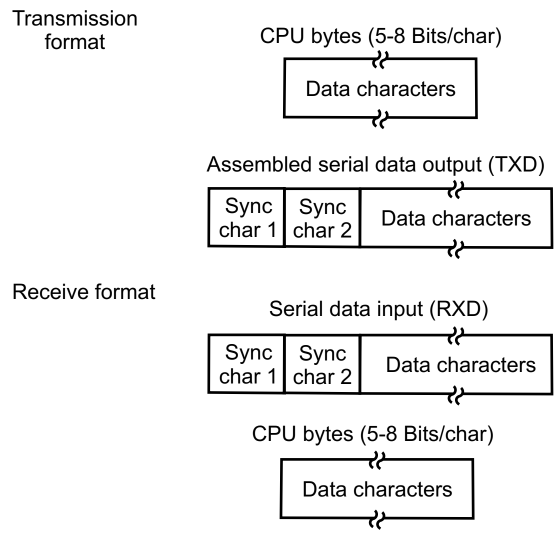


Fig. 9i.11: Synchronous mode transmission and reception (*Source:* Intel Corporation)

In the transmission format, either one or two synchronous characters are sent, followed by data characters. The number of synchronous characters (i.e., 1 or 2) is previously decided by the bit D_6 in the mode instruction format for synchronous operation. For such communication to take place, $C/\overline{D}$ will have to be 1. The characters are shifted out of the serial output register on the falling edge of the $\overline{TXC}$ (transmitter clock), and at the same rate as the $\overline{TXC}$. Once transmission commences, it is the duty of the CPU to replenish the transmitter buffer register in response to TXRdy. If the CPU fails to provide a character before the transmitter buffer becomes empty, 8251 automatically sends SYNC character(s). In such a case, TXE (Transmitter Empty) pin becomes high to indicate that the transmitter buffer is empty.

In the receive format, $C/\overline{D}$ is maintained at high level. In the internal SYNC mode, the receiver samples the data available as the RXD pin on the rising edge of $\overline{RXC}$. The command word should be previously programmed with the 'ENTER HUNT' command (bit D_7 of the Command Instruction Format) in the Enabled Condition (D_7 = 1). The receiver buffer register content is compared at every bit boundary with the SYNC character (previously loaded) till a match occurs. The process is extended to two SYNC characters if the 8251 is initially programmed for two SYNC characters (bit D_7 of the Mode Instruction Format). After 'HUNTING' is over, the system goes for character boundary synchronisation so that it can assemble the serial data to be subsequently changed to parallel format. The SYNDET pin is set high, which can be ascertained with a status read. This is resetted once status read is over. The SYNDET pin gets set in the middle of the parity bit if the parity is enabled; otherwise in the middle of the last data bit. In the external SYNC mode, 8251 comes out of HUNT mode by a high level on the SYNDET pin, which acts as an input in such a case.

26. Show the Command Instruction Format and explain the same.

Ans. The Command Instruction Format is shown below:

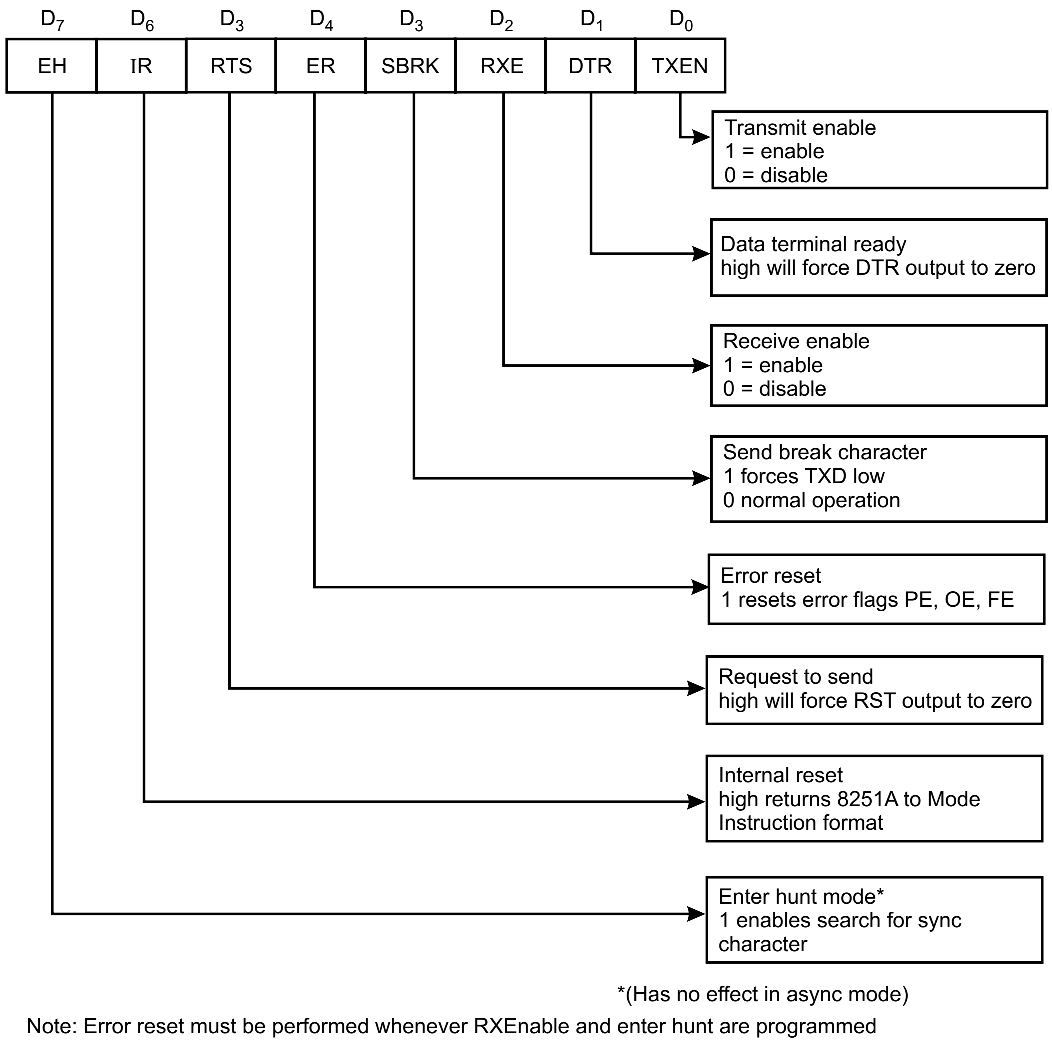


Fig. 9i.12: Command instruction format (*Source:* Intel Corporation)

The command instruction format controls the functioning of 8251. A command word with $D_6 = 1$ returns 8251 in mode instruction format. If D_0 (TXEN) is mode high, data transmission is possible whereas making D_2 (RXE) high, enables the system for reception. If D_1 (DTR) is made high, the DTR output will be forced in the zero state. A high on D_4 (ER) forces resetting of error flags PE, OE and FE (Parity, overrun and Framing errors respectively) in the status word. A high on D_3 (SBRK) forces TXD low while a zero corresponds to normal operation.

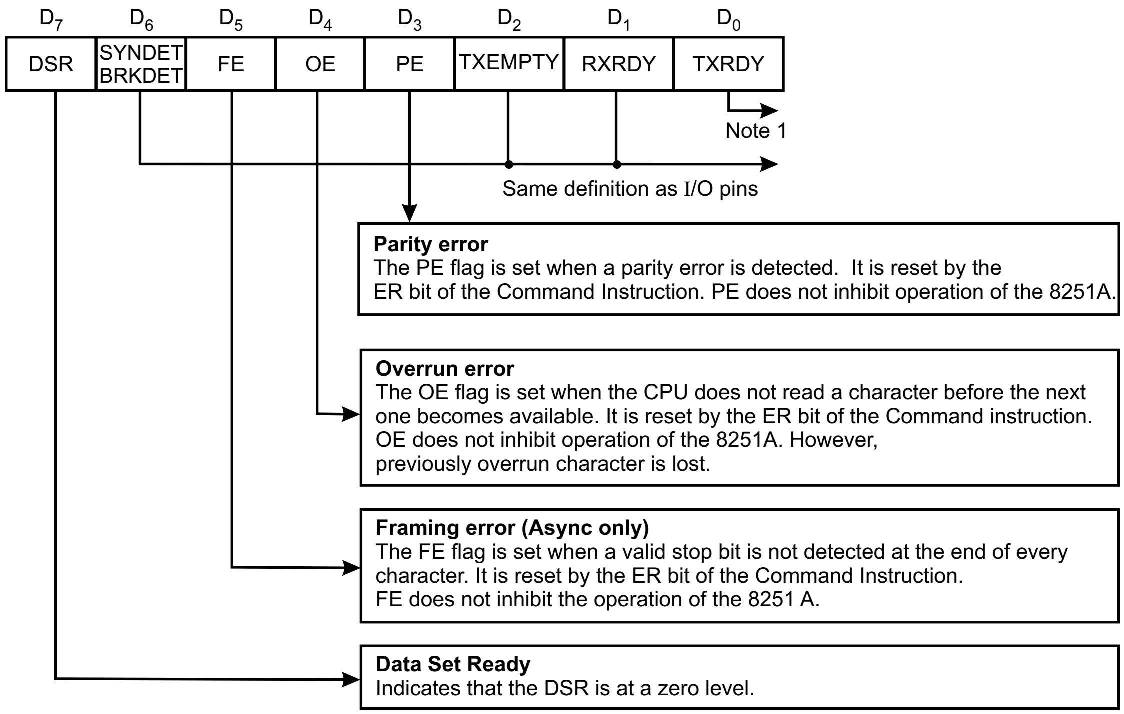
27. What happens when (*a*) power is switched on (*b*) the system is resetted?

Ans. On powering on the system, 8251 either enters into SYNC or command instruction format.

On resetting the system, 8251 returns to the mode instruction format from the command instruction format.

28. Draw the status word format and explain the same.

Ans. The status word format for 8251 is shown below:



Note 1: TXRDY status bit has different meaning from the TXRDY output pin. The former is not conditioned by $\overline{CTS}$ and TXEN; the latter is conditioned by both $\overline{CTS}$ and TXEN
i.e., TXRDY status bit = DB buffer empty
TXRDY pin out = DB buffer Empty. ($\overline{CTS}$ = 0). (TXEN = 1)

Fig. 9i.13: Status Word Format (*Source:* Intel Corporation)

The status word can be read with $C/\overline{D} = 1$. The CPU, for its proper operation, needs various informations. These are provided by the status word. It should be borne in mind that the status word is continuously updated by 8251, but not while the CPU reads it.

29. What are the modem control pins associated with 8251? Describe the functioning of these pins.

Ans. The modem control section of 8251 are handled by these four pins: $\overline{DSR}$, $\overline{DTR}$, $\overline{CTS}$ and $\overline{RTS}$. Out of these, the first and third are input pins (input to 8251) and the rest two are output pins. All these pins are active low. The signals on these pins are also used for purposes other than modem control. The description of these pins are given below:

$\overline{DSR}$ (Data Set Ready): This is a 1-bit inverting input port. It is used by the modem to signal the 8251 (here DTE) that it (modem) is ready to accept data for transmission. The DSR bit is checked by reading (polling) the D_7 bit of the status word. If it is low, then the modem can send data to 8251.

$\overline{\text{DTR}}$ **(Data Terminal Ready):** This is a 1-bit inverting output port. It is used by 8251 to signal the modem about its readiness to accept/transmit data. D_1 bit of command instruction word can either be set/reset, with a high D_1 bit forcing DTR output to zero.

$\overline{\textbf{RTS}}$ **(Request to Send):** This is a 1-bit inverting output port. It is used by 8251 to signal the modem that it has data to send. Bit D_5 of the Command Instruction Format controls the status of this pin.

$\overline{\textbf{CTS}}$ **(Clear to Send):** This is a 1-bit inverting input port. It is used by modem to signal 8251 that it has the right of way over the communication channel and can send out serial data. Bit D_0 of the command instruction word should be enabled for the above to be realised.

If D_0 is made low in command instruction word while data transmission is taking place or if $\overline{\text{CTS}}$ is switched off, the transmitter will complete sending the data stored in its buffer prior to getting disabled.

30. What is the baud rate of 8251?

Ans. The asynchronous baud rate of 8251 is 9600, while for the improved version of 8251—i.e., 8251A, this is 19, 200.

31. Discuss how a noise pulse may be recognised as a valid start pulse. How this possibility is eliminated?

Ans. In the asynchronous case, a USART may be programmed for receive clock rates of 8,16,32,64 times the receive data rate (these correspond to 8X, 16X, 32X and 64X). Thus the receive clock rates may be 8X RCP, 16X RCP, 32X RCP and 64X RCP, apart from the normal 1X RCP. Actually, 1X RCP corresponds to the receive data rate.

Fig. 9i.14(a) shows the situations when the line is idle (i.e., in state '1') and is hit by a noise impulse. The receive clock pulse (RCP) is set at 1X RCP. The figure shows the uneventful situation of the clock pulse sampling the input line at the instant the noise is present (point AA') and the circuit detects a low. This gives rise to an invalid start bit and the subsequent clocks will interprete the high condition on the data line to be data bits—all at logic 1's. This gives rise to a serious error arising out of an accidental noise pulse.

Fig. 9i.14(b) shows the same situation with the exception that the receiver clock is now made sixteen times faster—i.e., 16X RCP. Once a low is detected, the receiver is made to wait for seven clock cycles before it resamples the input data (this corresponds to BB' in the fig.). Since in this case the receiver analyses the input line status to be '1', hence it concludes that the low input line status that it detected seven clock cycles earlier to be a noise pulse. Thus the possibility of the UART receiver accepting spurious noise pulse is eliminated. This can further be improved by increasing the clock rate to 32X, 64X, etc.

Fig. 9i.14(c) shows the input line scanned by the same 16X RCP. It shows a valid start bit followed by data bits. As in Fig. (b), here also the receiver waits for seven clock cycles after detecting a low. Here, the receiver detects a low for the second time and comes to the conclusion that a valid start bit has arrived. Thus a valid start bit is detected at CC'. Thereafter the input data is sampled once every 16 clock cycles—this makes the sample rate equal to the receive data rate. This way the stop bit is detected and immediately the receiver goes into start bit verification mode.

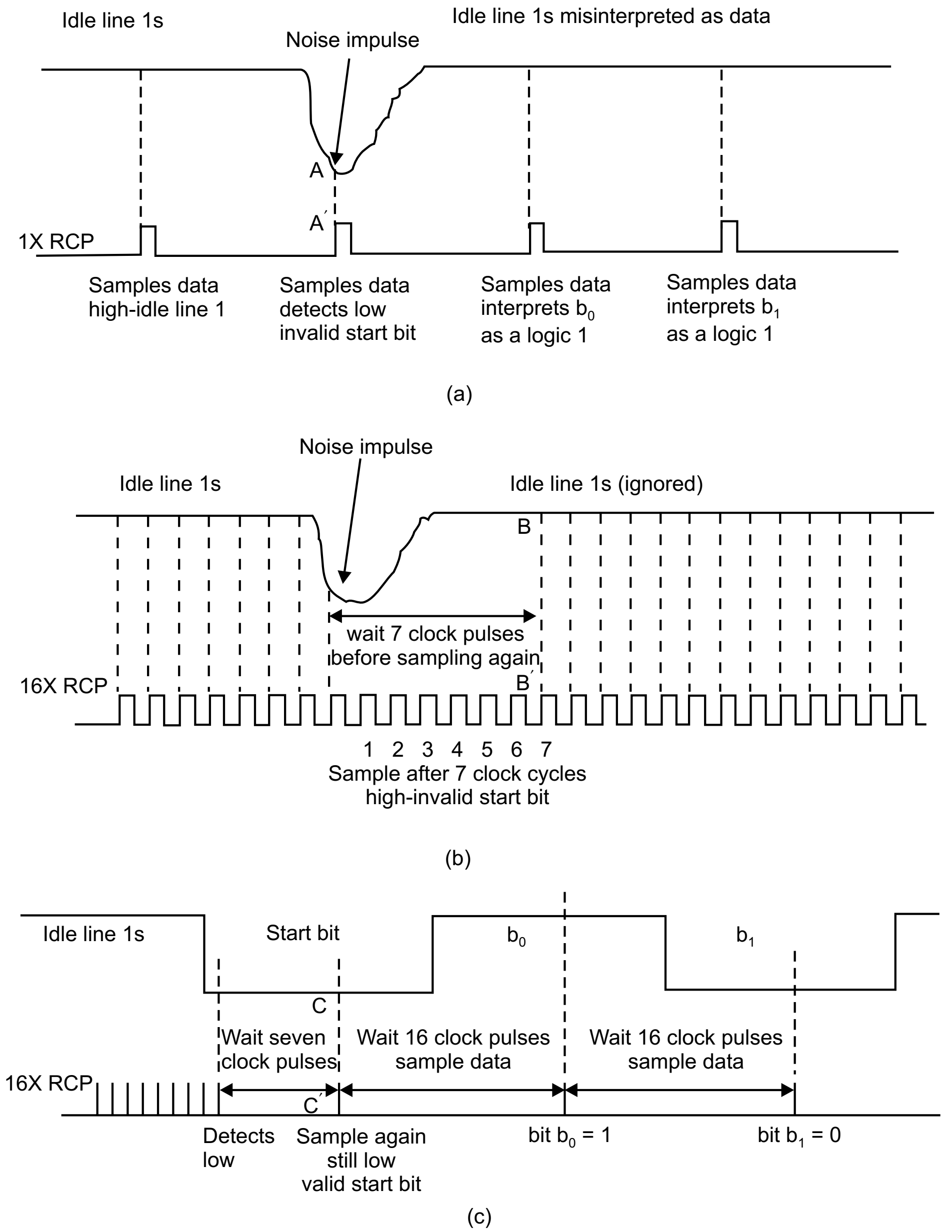


Fig. 9i.14: Start bit verification (a) 1X RCP, (b) 16 X RCP; (c) valid start bit.

32. Explain detection error and sampling error.

Ans. The situation is explained in Figure 9i.15 by clocking the UART receiver with 16X.

The difference in time between the beginning of a start bit and its detection is called detection error and is shown by t_d in the figure. The maximum time of detection is one RCP.

The difference in time between when a sample is taken (i.e., put into the receive shift register) and the actual centre of a data bit is called the sampling error and is shown by t_s in Fig. 9i.15.

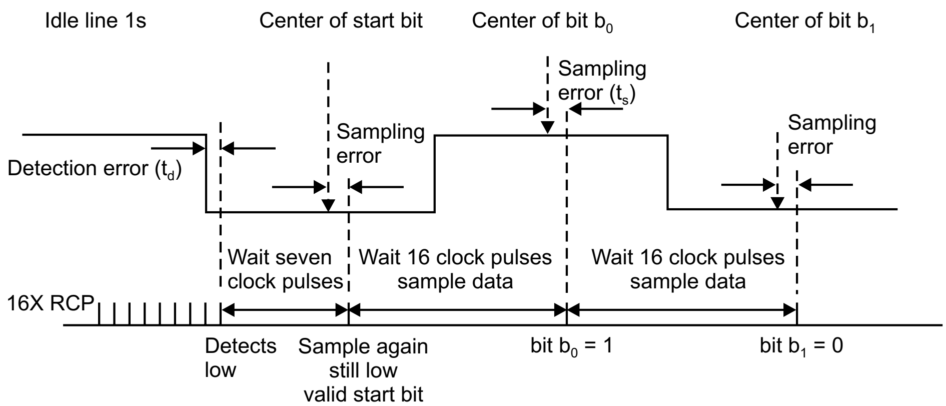


Fig. 9i.15: 16X receive clock rate.

33. What happens to the maximum detection error when receive clock rate equals the receive data rate?

Ans. The maximum detection error would approach one bit time. Thus a start bit would not be detected until the very end of the start bit.

34. What is meant by clock slippage?

Ans. Clock slippage, also known by the name of clock skew, is a problem faced in asynchronous communication system.

In this case, the magnitude of sampling error increases with each successive sample in the data bit pattern. Thus, the clock may slip over or slip under the data.

Sometimes it may so happen that a data bit (it would start occurring for latter data bits in the data stream) may be sampled twice or not sampled at all in the clock period—it depends on whether the receive clock is higher or lower the transmit clock.

35. How the sampling error is related to sampling rate?

Ans. As the sampling rate is continued to be increased, the sampling error goes on decreasing. As the sampling rate is increased, the sample time moves closer and closer to the centre of data bit, thereby decreasing the sampling error.

10. BUS STANDARDS

10a

RS:232C Standard

1. What logic convention is followed in RS-232C and indicate the corresponding voltage levels also.

Ans. A negative logic convention is used for such a standard. Logical '1' is represented by a transmitted voltage level in the range of –3V to –15V, while a logical '0' is represented by a transmitted voltage level in the range of +3V to +15V.

2. What is the noise margin of RS-232C and how does it compare with the noise margin of TTL ICs?

Ans. The noise margin of RS-232C is 2V while that for TTL, it is 0.4V only.

The higher noise margin level of RS-232C allows it to pass through more noisy levels than that permitted for TTL.

3. How RS-232C is interfaced with TTL?

Ans. For such an interface to be done, line drivers and receivers are required. MC 1488 accepts TTL level inputs and provides an output compatible to RS-232C. IC MC 1489 converts RS-232C levels to TTL levels.

4. What is the transition time allowed for RS-232C?

Ans. The transition time (time allowed to switch over from one voltage level to another voltage) is 4% of one bit time. Thus at 19,200 baud, the transition time is $0.04 \times \frac{1}{19,200}$ second $= 2.1\ \mu S$.

5. What restriction does the transition time impose on the maximum cable length that can be used with RS-232C.

Ans. The transition time puts a limit on the maximum length of cable for information transformation on the RS-232C cable. As cable length increases, so also the capacitive load. This thus impedes transition time. For a baud of 19,200, the maximum cable length becomes around 5 ft.

6. Mention in which case RS-232C is used.

Ans. It is used for serial asynchronous data transmission (*a*) over telephone lines equipped with modems, (*b*) in digital systems in which the total cable distance is less than 50 ft.

7. How many pins are there on a RS-232C connector? In how many ways are they divided?

Ans. There are 25 pins on a RS-232C connector. They are divided into two groups—three (3)

wires are used for data transmission/reception purposes. Pin 2 is used for Transmitted Serial Data (TXD), pin 3 for Received Serial Data (RXD) and pin 7 for Signal Ground.

In the other group, twenty-two (22) wires are there which are used for control purposes.

The following figure shows the connections between DTE and DCE for data transmission purposes.

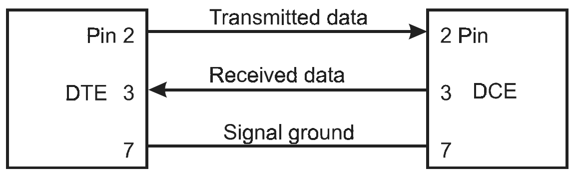


Fig. 10a.1: RS-232C uses three wires for data transmission between DTE and DCE. All other wires associated with the interconnection are control signal lines

The pin assignments of RS-232C are as here under:

1. Protective (chassis) Ground
2. Transmitted Serial data (TXD)
3. Received Serial data (RXD)
4. Request to Send (RTS)
5. Clear to Send (CTS)
6. Data Set Ready (DSR)
7. Signal Ground
8. Received Line Signal Detector (DCD)
9. (Used for Data set testing)
10. (Used for Data set testing)
11. Not Used
12. Secondary Received Line Signal detector
13. Secondary Clear to Send
14. Secondary Transmitted Data
15. Transmission Signal Element Timing (DCE Source)
16. Secondary Received Data
17. Receiver Signal Element Timing
18. Not Used
19. Secondary Request to Send
20. Data Terminal Ready
21. Signal Quality Detector
22. Ring Indicator
23. Data Signal Rate Selector (DTE/DCE Source)
24. Transmit Signal Element Timing (DTE Source)
25. Not Used

8. Indicate to what extent the 50 ft distance limitation of RS-232C can be extended.

Ans. It can be extended up to 10,000 ft by using line drivers and twisted pair cables, but at the expense of reduced baud—600 only. Line drivers are signal converters and amplify the digital signals but don't modulate them into audio signals. This use of line drivers don't change the transmission frequency also–thus it is referred to as base band signalling.

10b

IEEE:488 Bus

1. What is a IEEE-488 Bus?

Ans. It is a byte serial, 8-bit parallel, asynchronous type of instrument interface and was introduced by IEEE in 1975. It comes in a standard 24-pin connector.

2. What are the other names by which IEEE-488 bus is known?

Ans. The other names of IEEE-488 bus are GPIB (General Purpose Interface Bus) and the HP (Hewlett-Packard) Interface bus.

3. How the IEEE-488 bus evolved over time?

Ans. The GPIB (General Purpose Interface Bus) was originally developed by Hewlett Packard in 1965—that time called HPIB and was used as a communications standard to connect and control programmable instruments.

IEEE in 1975 introduced a standard, known as, IEEE standard 488-1975 (or IEEE-488) for high speed data communication between instruments manufactured by different companies. The current version of the same standard is referred to as IEEE standard 488.1-1987 and defines the mechanical, electrical and hardware protocol specifications of the communications interface but did not address data formats, message exchange protocols, common configurations commands, device-specific commands, status reporting, error handling, etc.

In 1987, IEEE introduced another standard—called IEEE standard 488.2. This standard is fully compatiable with the earlier standard 488.1 and addresses software protocol issues elaborately.

SCPI (Standard Commands for Programmable Instruments) was introduced in 1990 by several manufacturers and is an improvement over IEEE 488.2 standard. SCPI utilises IEEE 488.2 as a basis and defines a common command set for programmable instruments interconnected with varied hardware links.

4. Write down the advantages of IEEE-488 based measurement system.

Ans. Advantages which are derived by using an IEEE-488 based measurement system are :

- High measurement throughput—ranging from 10 to 100 times faster than conventional manual methods.
- Stored data can be analysed/processed as per the requirements of the particular situation adding functionality to the system.
- Repetitive manual operations can be dispensed with.
- Settings of equipments are highly repeatable—resulting in high consistency in measurements.

- Less skilled workers can handle the system.
- Very high consistency in measurements.

5. What is the maximum data transmission rate of IEEE-488 bus?

Ans. The maximum data transmission rate is 1 mega byte/second. But actual data rate is governed by instruments connected to the bus, as also the distance involved.

6. How many instruments can be connected to such a bus?

Ans. A maximum of 15 instruments can be interconnected to such a bus, but in such a case the total cable length does not exceed 20 m (66 ft).

7. What logic convention is followed and what logic family it is compatiable to?

Ans. Signals transmitted over the IEEE-488 bus are TTL logic compatible.

It follows a negative logic convention with logic 0 = TTL high state (≥ 2.0 V), and logic 1 = TTL low state (≤ 0.8 V).

8. Distinguish between the IEEE-488 bus standard and the IEEE-488 bus.

Ans. The IEEE-488 standard is a document that indicates the rules, specifications, timing relationships, physical characteristics, etc. that the instruments connected to the bus must adhere to. This standard includes electrical, mechanical, functional specifications of the instruments interface. The electrical specifications include the current and voltage levels of the transmitted signals, mechanical specifications tell about the number of wires, type of connector, pin designations, etc. The functional specifications indicate the functioning of the different lines, protocols to be obeyed for message transfer, timing relationship between the signal lines and the different kinds of messages that can be carried between different devices.

On the other hand, the IEEE-488 bus is the hardware that is actually used to implement the standard.

9. How many lines comprise the IEEE-488 bus? Discuss.

Ans. There are in all sixteen (16) lines that comprise the complete IEEE-488 bus.

These 16 lines are divided into three categories—data lines (8 nos.), handshake lines (3 nos.) and bus management lines (5 nos.). This is shown Fig. 10b.1.

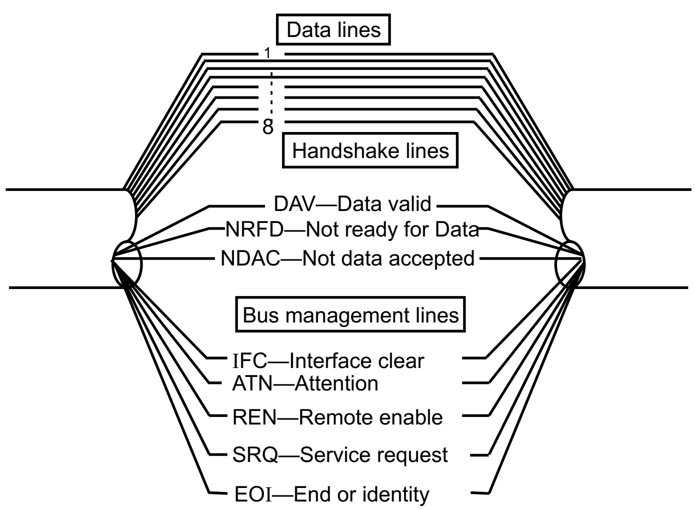


Fig. 10b.1: Sixteen lines make up the complete IEEE-488 bus structure

The data lines carry measurement data, program data, addresses, universal commands etc. Handshake signals are required because of asynchronous nature of operation of the bus, whereas the management lines are employed to ensure an orderly flow of data across the bus interface.

10. Is it possible to remove the 20 m restriction while using IEEE-488 bus?

Ans. It is possibly by using 'bus extenders'. With bus extenders, an IEEE-488 bus system can be extended to several thousand miles. In such a case the instruments are categorised into two groups—remote and local. Functionally there is no difference between a remote instrument and a local instrument connected to the bus—thus the extenders are essentially transparent in nature.

For distances up to 1000 m, several bus extenders are employed by using twin-pair cables in full duplex form.

If it is beyond 1000 m, then RS-232C cable is employed by serialising the information available on the 16 lines.

Maximum transmission bit rate is decreased by using bus extenders. This value is kept at 20 K bits/second for transmission up to 1000 m.

Bus extenders support synchronous and asynchronous modems when transmission involves huge length that includes telephone lines. In such a case, 19.2 K bits/second (synchronous) and 150, 300, 600 or 1200 bits/second (asynchronous) transmission rates are achieved.

11. In how many categories the instruments connected to the IEEE bus can be grouped ? Discuss each such category.

Ans. There are four categories of instruments which are connected to the IEEE bus. These are controllers, talkers, listeners and talkers/listeners. Such a connection is shown in Fig. 10b.2.

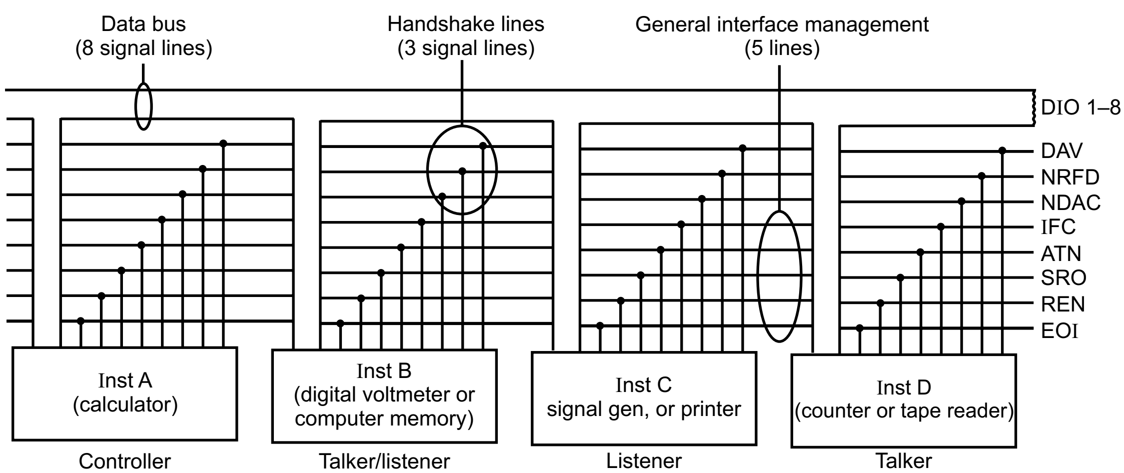


Fig. 10b.2: Block diagram of an IEEE-488 bus connected system

Controllers (it may be a computer or a programmable calculator) manage all the operations of the instruments connected to it and direct data flow from one to another.

Listeners are those instruments which receive data. Examples of listeners may be a printer or a recorder.

Talkers send data to listeners. Example of a talker is a digital voltmeter.

Talkers/listeners are able to receive or send data. These may be digital multimeters and network analysers.

12. How many talkers and listeners can be active at any given instant of time?

Ans. At any given instant of time, there can be only one talker (the one which sends data) but there can be more than one listener (the one which receives data).

13. What determines the data transmission rate on the IEEE bus?

Ans. When there are more than one listener who receive data, it is the slowest of them all which determines the data transmission rate.

14. What makes a listener or a talker to become 'active'?

Ans. Each of the talkers or listeners has a unique address associated with it. A talker or a listener can become active by invoking the corresponding unique address.

15. Who controls the three handshake lines?

Ans. The three handshake lines are DAV (data valid), NRFD (not ready for data) and NDAC (Not data accepted).

It is the active talker and listeners who control the three handshake signals. The particular talker and listener are made 'active' by the controller, after which the controller has no part to play until data transfer is over.

DAV line is controlled by the active talker which places valid data on the data lines. The NRFD and NDAC lines are controlled by the active listeners to indicate their preparedness to receive data and acceptance of data respectively.

16. Discuss the function of the GPIB controller.

Ans. At any given point of time, only one device can act as an active controller. Another device may seek the control and can act as a controller once the active controller passes control to it.

For proper communication, each device is assigned a unique GPIB address—called primary address of the device. The primary address ranges from 0 to 30, although a maximum of 15 devices can be connected via the bus. Assignment of a device address is by DIP switches, jumpers, etc.

Usually, a PC acts as a controller—a GPIB control card is thus installed in the PC then. This controller designates which device will be talker and which device(s) would be listener(s). The controller, during initialisation, assigns the device addresses.

A new GPIB protocol—called HSS 488, introduced by National Instruments, allows data transfer rates up to 8 MB/second.

17. How the devices are configured via GPIB?

Ans. The devices, connected via the GPIB bus, are configured in one of the three ways :

- Star
- linear (chain)
- a combination of the two.

Figure 10b.3. shows the star and linear configurations. For star connection, all connectors are connected to the same port of the controller. The connected devices must be physically close to the other controller because of length limitation of each cable imposed by the standard.

In linear or chain configuration, each device, including the controller, is connected with the next one on the chain. The controller can be placed anywhere in the chain. In this, the software needs reconfiguring in case a device (including its cable) is removed.

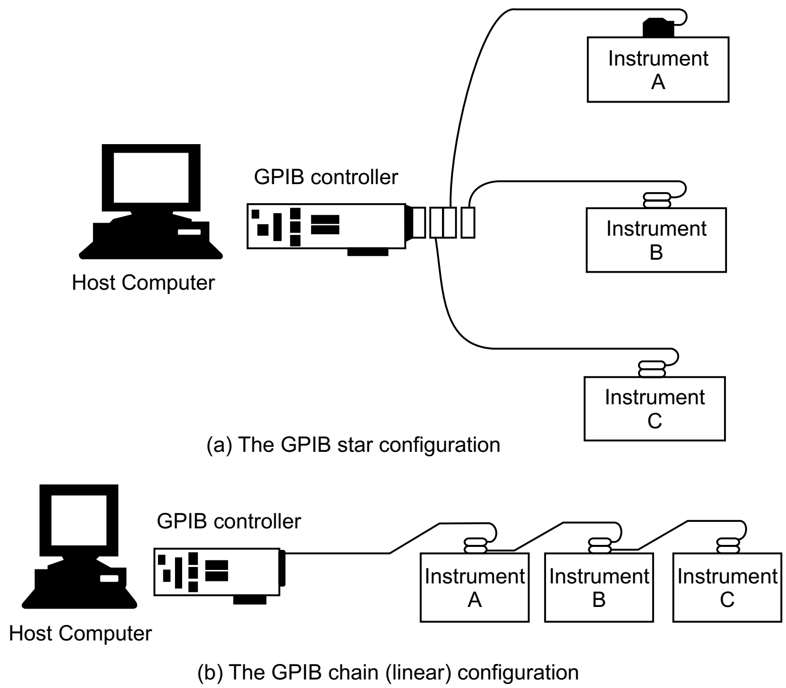


Fig. 10b.3: GPIB configurations (a) star (b) linear (chain)

18. Draw the IEEE-488 bus connector and state the functions of each pin.

Ans. The IEEE-488 bus that connects the various instruments and controllers are connected via a 24 pin IEEE-488 bus connector. The connector is a 24 pin type, shown in figure 10b.4.

The pin numbers, their abbreviations and corresponding functions are tabulated in Table 10b.1.

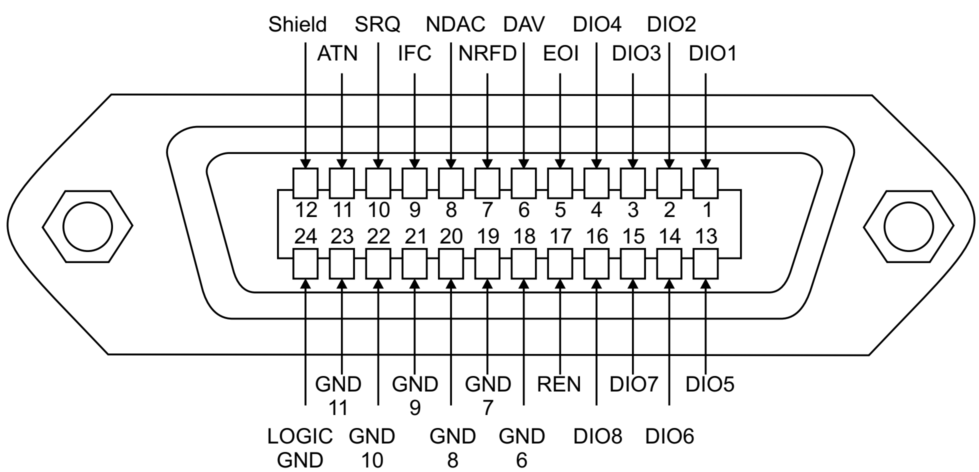


Fig. 10b.4: IEEE-488 bus connector

Table 10b.1: Connector pin details

Pin number	Abbreviation	Function
1	DIO1	Data line 1
2	DIO2	Data line 2
3	DIO3	Data line 3
4	DIO4	Data line 4
5	EOI	End of identify. The signal is generated by a talker to indicate the last byte of data in a multi-byte data transfer. EOI is also issued by the active controller to perform a parallel poll by simultaneously asserting EOI and ATN.
6	DAV	Data valid. This signal is asserted by a talker to indicate that valid data has been placed on the bus.
7	NRFD	Not ready for data. This signal is asserted by a listener to indicate that it is not yet ready to accept data.
8	NDAC	Not data accepted. This signal is asserted by a listener whilst data is being accepted. When several devices are simultaneously listening, each device releases this line at its own rate (the slowest device will be the last to release the line).
9	IFC	Interface clear. Asserted by the controller in order to initialize the system in a known state.
10	SRQ	Service request. This signal is asserted by a device wishing to gain the attention of the controller. This line is wire–OR'd.
11	ATN	Attention. Asserted by the controller when placing a command on to the bus. When the line is asserted this indicates that the information placed by the controller on the data lines is to be interpreted as a command. When it is not asserted, information placed on the data lines by the controller must be interpreted as data. ATN is always driven by the active controller.
12	SHIELD	Shield
13	DIO5	Data line 5
14	DIO6	Data line 6
15	DIO7	Data line 7
16	DIO8	Data line 8
17	REN	Remote enable. This line is used to enable or disable bus control (thus permitting an instrument to be controlled from its own front panel rather than from the bus).
18–24	GND	Ground/common signal return.

Notes :

1. Handshake signals (DAV, NRFD and NDAC) employ active low open-collector outputs which may be used in a wired-OR configuration.

2. All remaining signals are fully TTL compatible and are active low (asserted low).
3. Pins 18 to 23 are intended for use with twisted pair grounds for the control signals (DAV, NRFD, etc.) that appear on pins 6 to 11 on the other side of the connector).

19. How the request for service is taken care of by the system?

Ans. It is the SRQ (Service request) line which gets asserted when a device requests for service from the controller. The SRQ line is a wire-ORed, i.e., the device service lines are ORed together and connected to the SRQ pin of the bus connector. The device which has interrupted is identified by the controller by polling schemes—either serial or parallel.

For serial polling, each device places a status byte on the bus with DIO7 set if the device in question is requesting service, otherwise it will be in reset condition. Now the active controller polls each device to get to know the particular device which has generated the service request. The remaining 7 bits of the status byte represent the status of the particular device.

For parallel polling an individual data line is asserted by each device. Thus it is very easy for the controller to determine/identify the device which has drawn the attention of the controller.

20. Mention the services provided by GPIB bus.

Ans. Operations available on the GPIB bus include:

- to support request of service from a device
- clearing devices selectively or universally
- device dependent functions.

When a device seeks to get the service from the controller, it asserts for the same via the SRQ line. The controller, on its part, determines the particular device via either serial or else parallel polling scheme.

In the serial polling scheme, the controller, via a command, enquires about the status of each device one often another. This scheme is a slow one in which the last device which is queried, has the least priority.

In the parallel polling scheme, the command issued by the controller, polls all the devices connected to the system simultaneously. It is a fast process but a complicated one.

21. Show the interface capabilities of GPIB, their functions and also their descriptions.

Ans. The interface capabilities of GPIB, their symbols and also their functions are tabulated below, vide Table 10b.2.

Table 10b.2: GPIB capabilities, symbols and functions

Interface Capability		
Symbol	**Function**	**Description**
T	Talker	Capability of a device to talk.
TE	Extended Talker	Capability of a talker to extended addressing.
L	Listener	Capability of a device to listen.
LE	Extended Listener	Capability of a listener to extended addressing.
SH	Source Handshake	Capability of a device to send command or data bytes (multiline message) over GPIB using the three wire handshake protocol.

AH	Acceptor Handshake	Capability of a device to receive command or data bytes (multiline message) from GPIB using the three wire handshake protocol.
RL	Remote/Local	Capability of the device to switch between response to its front panel controls and response to commands sent over GPIB.
SR	Service Request	Capability of a device to assert service request to obtain service from controller.
PP	Parallel Poll	Capability of a device to indicate that it needs service when the controller requests the status.
DC	Device Clear	Capability of a device to be reset.
DT	Device Trigger	Capability of a device to be triggered by a GPIB command to perform a device dependent function.
C	Controller	Capability of a device to send addresses, universal commands and addressed commands to other devices on the bus. It also includes the capability of the device to conduct polling to determine a device requesting a service.
E	Drivers	It describes the types of electrical bus drivers used in the device.

22. Discuss the GPIB commands.

Ans. The GPIB commands are classified into several groups.

- Talk address commands
- Listen address commands
- Universal commands
- Addressed commands
- Secondary commands

The first four fall under the purview of Primary commands. All the functions under these four commands are shown in Table 10b.3.

Table 10b.3: The various GPIB commands

No.	Command		Code		Function
	Symbol	**Name**	**Binary**	**Hex**	
	Talk address commands				
1.	TAD	Talk Address	010xxxxx	40H+addr	Selects device with address 'xxxxx' to talk
	MTA	My Talk Address			
2.	UNT	Untalk	01011111	5FH	Deactivates talker
	Listen address commands				
3.	LAD	Listen Address	001xxxxx	20H+addr	Selects device with address 'xxxxx' to listen
	MLA	My Listen Address			

4.	UNL	Unlisten	00111111	3FH	Deactivates listener
	Universal commands				
5.	LLO	Local Lock Out	00010001	11H	Prevents manual settings of front panel controls
6.	DCL	Device Clear	00010100	14H	Initializes all devices to power-on condition
7.	PPU	Parallel Poll Unconfigure	00010101	15H	Disables parallel poll response
8.	SPE	Serial Poll Enable	00011000	18H	Enables to output status byte when talk
9.	SPD	Serial Poll Disable	00011001	19H	Disables serial poll response
	Addressed Commands				
10.	GET	Group Execute Trigger	0000 1000	08H	Signals a group of devices to begin executing the triggered action.
11.	GTL	Go to Local	0000 0001	01H	Puts the selected device in local mode
12.	PPC	Parallel Poll Configure	0000 0101	05H	It instructs addressed listeners to interprete the byte that follows the command as secondary command rather than a secondary address. To enable parallel poll, PPE (Parallel Poll Enable) follows PPC and to disable parallel poll PPD (Parallel Poll Disable) follows PPC
13.	SDC	Selected Device Clear	0000 0100	04H	It resets the addressed listeners to a device dependent state
14.	TCT	Take Control	0000 1001	09H	It instructs the active talker to become active controller.

Secondary commands, when used, are always used in conjunction with primary commands. When the number of devices to be addressed exceeds 30, an extra byte is needed to specify these devices for their talk/listen functions. This extra byte is the secondary address command byte. This is sent immediately after talk/listen address command. The secondary address command is 60H+address. The maximum address limit is 961 by this method. This type of addressing is called extended addressing and the devices are called extended talkers/listeners.

23. Discuss the End of Interface Message/Data.

Ans. There are three ways to achieve the above. These are:

- EOS (End of String)
- EOI (End of Identity) and
- Count Method

Talkers/listeners must be configured to use one of the three methods to achieve the above—the manual must be consulted to configure in one of the three methods.

EOS uses an EOS character like carriage return (OD H) or a new line (OA H). A talker places the EOS character at the end of the data string. The listeners read the character bytes until it comes across the EOS character—then the listener understands that no more data bytes are there and the listener terminates the read operation.

EOI method uses the EOI signal existing as the GPIB signal line to terminate the message. The talker sets the EOI line status high at the end of message transmission. The listener detects the active EOI signal (high) and understands that no more data bytes are there from the talker. The listener then terminates the read operations.

In the third i.e., count method, the device receives the information about the total number of bytes to be read. Thus in this method the talker can send a specified number of bytes on the bus.

In some cases more than one combination is used for termination of message. In such a case, the termination is effected on the basis of logical ORing of the methods employed.

24. How GPIB programming is done?

Ans. A cluster of devices/instruments connected via the GPIB bus share data/information between them. The GPIB bus helps establish this sharing of information, perform management and bus control operations, sending device dependent/independent commands, sending data either in string or binary format.

GPIB programming is a must to achieve all of the above—the programming can be done either in low level or high level versions.

In the low level programming, the GPIB controller sends the commands which are necessary for bus management and addressing in sequence the devices which are connected via the bus. In high level programming, the overhead due to codes required for programming is drastically reduced. In this, a programmer needs to know lesser information about GPIB protocol and bus management operations.

High level language involves four methods—DOS device driver, onboard EPROM, program language interface and Windows DLL.

25. What are the jobs performed by the five management lines?

Ans. The five management lines are : IFC (Interface Clear), ATN (Attention), REN (Remote Enable), SRR (Service request) and EOI (End or Identity).

The jobs performed by the IFC line are:

(a) It is used to initialize the bus and clear it if something goes wrong.

(b) It helps return the bus in a quiescent or known state. This is done by the controller to override all current activities and abort all present data transfer processes.

The jobs performed by the ATN line are:

(a) How many data lines are being processed at any given instant of time.

(b) When ATN line is true, data lines contain either addresses or universal commands. In such a case only the controller talks.

(c) When ATN line is false, the addressed devices can use the data line. In such a case, data that are transmitted are device dependent.

The jobs performed by the REN line are:

(a) When REN is active, it allows the bus to control a device.

The jobs performed by the SRQ line are:

(a) With its help, a device can seek the attention of the controller. Such a case arises when the device has data to send (the device is a talker) or the device is to receive data (the device is a listener).

(b) Such a request (and not a command) can be rejected by the controller if it does not have any time to service the request.

The jobs performed by the EOI line are:

(a) It can be used by the controller as a polling line.

(b) It can be asserted by the active talker to designate the end of a message.

26. Draw and explain the timing diagram showing the sequence for data transfer to take place.

Ans. The following figure shows how data transfer is effected taking the help of handshake signals.

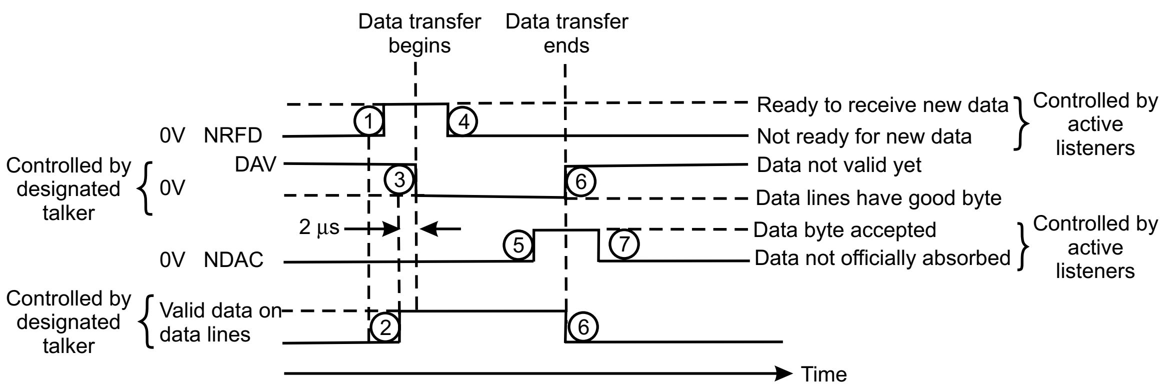


Fig. 10b.5: Timing diagram showing the sequence of events during an IEEE-488 bus handshake

Sequence of events that follow on the time scale (1) ⇒ (2) ⇒ (3) ⇒ (4) ⇒ (5) ⇒ (6) ⇒ (7).

The sequence of events that take place can be explained as follows:

1. Listener(s) which will accept data indicate their readiness with regard to their data acceptance via NRFD (Not Ready For Data) line. If NRFD is low, it indicates that the listener(s) is/are not ready. Only when all the listeners are ready, the NRFD line will go high.
2. The designated talker then drives all the eight input/output lines, causing valid data to be placed on them.
3. The active talker pulls the DAV (data valid) line to zero volts 2 µS after serial number (2). This 2 µS is required for the data to settle to their respective values on the data lines.
4. The listeners now pull the NRFD line to zero, which prevents any additional data transfer from being initiated. The listeners then accept data at their individual rates.
5. When all the listeners have accepted data, the NDAC line goes high.
6. When the active talker sees that NDAC line has become high, it stops driving the line and also it releases the DAV line. The talker is now in a position to put the next data on the data bus.

7. NDAC line is pulled down to zero volt by the listeners and the data is then taken 'away' by the listeners.
8. Thus the sequence of operations that take place on the time scale are
   (1) $\Rightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (4) $\Rightarrow$ (5) $\Rightarrow$ (6) $\Rightarrow$ (7) and back to (1) again.

10c

The Universal Serial Bus (USB)

1. Mention the data transfer rates supported by USB.

Ans. USB supports three types of data transfer rates
480 Mbps (high speed)
12 Mbps (full speed)
1.5 Mbps (low speed)

2. What are the units in a USB system?

Ans. There are three units in a USB system—a USB host, a USB device and the USB cable. A PC acts as a USB host, a scanner or printer acts as a USB device. The host and the device are connected by the third unit that comprise the USB system—USB cable.

3. Mention some of the application areas of USB.

Ans. Depending on the data transfer rates, the universal serial bus supports a host of application areas.

Keyboards, joysticks, mice, etc. are some of the input devices that are supported by USB for low speed applications (up to 128 KB/sec.)

USB supports medium speed (128 KB/sec. to 2 MB/sec.) applications in the areas of low and high speed modems, process plant instrumentation, scanners, ZIP drives, sound cards, etc.

For high speed application (> 2MB/sec.) areas, examples are network adapters, optical drives, low bandwidth video, etc.

4. Discuss the features and advantages of universal serial bus.

Ans. Universal serial bus is a plug-and-play facility and connects a host computer to many simultaneously accessible peripheral devices. The resources are made available via USB through a host-scheduled, token-based protocol. While the host computer and peripheral are in operations, other peripherals can be attached, configured, used and detached via this bus which is mostly not possible with others forms of computer bus.

The characteristics associated with universal serial bus are:

- Master/slave, half-duplex, timed communication bus system and is designed to connect peripherals and external hubs.
- Master or host hub has complete control over transaction.
- Peripherals cannot initiate a communication on the USB bus.
- Supports data transfer rates up to 480 Mbps.
- Supports various types of data transfers.
- Supports real time data for voice, audio and compressed video.

- Ability to adapt to multifarious system configurations.
- Concurrent operation up to a maximum of 127 devices possible.
- Plug and play feature.
- Wide bandwidth with entire bus width can be used in case of isochronous mode.
- Peripheral devices can be 'hot-plugged' and 'hot-unplugged'.
- Device identification and configuration possible very easily.
- Cabling and connection procedure very simple.
- Easy to set up and configure.
- A host of data rates possible.
- Error detection and fault recovery possible.
- Flow control embedded into the protocol.
- Non-proprietary, open standard system.
- Can be connected from PII to higher end PCs.
- Up to 8 USB connectors can be provided for new PCs.

5. Draw the schematic of a modern PC motherboard with serial and parallel ports as well as USB ports.

Ans. In modern PC motherboards, USB ports are straightway available, while such facility is made available in older PCs by inserting USB adapter cards. Fig. 10c.1 shows the modern PC version of a motherboard which has the USB ports available from it.

Thus the kind of port facilities available from such a motherboard are : USB ports, Parallel ports LPT1 and serial (RS–232) ports COM1 and COM2. For implementing USB standards, PCs must have atleast Windows 98, 2000 or XP version facilities in it.

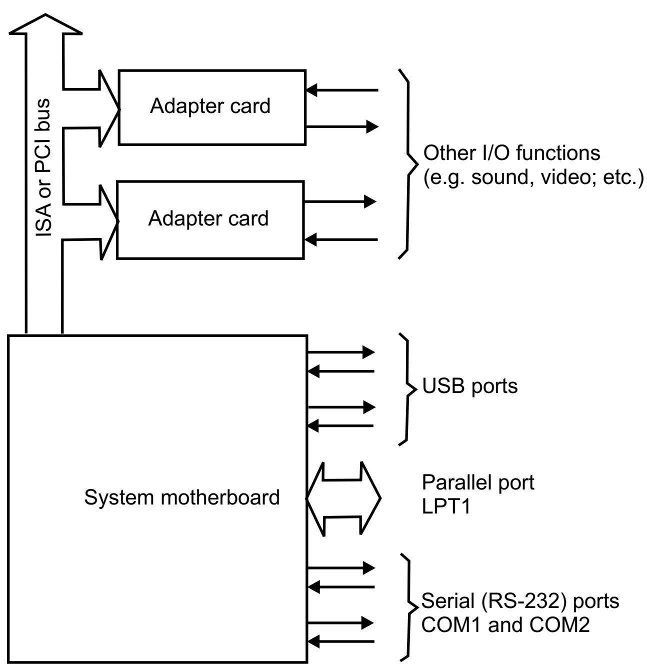


Fig. 10c.1: A modern PC with motherboard USB ports and host controller

6. Discuss (a) USB host (b) USB device.

Ans. (a) USB host:

A USB host is comprised of the following:

- USB host controller hardware
- USB system software
- Client software

USB system software comprises controller driver, OS and USB driver, while client software has the device driver for the USB device. The USB host:

- detects attachment/detachment of USB devices.
- manages data/control information flow either way between the host and the USB devices.
- keeps track of the status information of the USB devices.
- powers USB devices which don't have power of their own.

(b) USB device:

A USB device has the following:

- a USB bus interface hardware
- a USB logical device and
- functions, i.e., many capabilities that a USB device can provide.

Again the bus interface hardware comprises a transreceiver and a Serial Interface Engine (SIE). The former provides the electrical requirements while the latter is meant for providing the bit timings on the bus.

The functionalities of a USB device are:

- Responds to all host requests.
- Always monitors the device address in each communication and selects self when the device address matches with that sent by communication from the host.
- When it receives some data, it checks for errors in the error checking bits. If it detects errors, then it requests for retransmission of data. When the device sends data, check bits are added before sending.

7. Explain how USB interface is implemented and also indicate the USB data signals.

Ans. USB uses two lines for differential data connections (designated D+ and D–) and also two lines for supplying power to the cable—one is V_{BUS} (nominal value +5V) and the other customary GND wire.

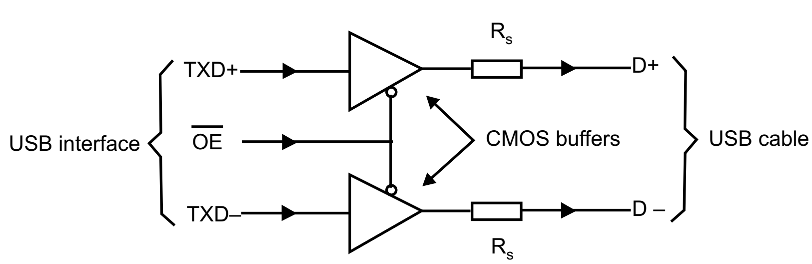


Fig. 10c.2: USB buffered interface

The USB buffered interface is shown in Fig. 10c.2.

Three lines are input to the CMOS buffers — T X D+, T X D– and $\overline{OE}$ (output enable). The data signal levels must conform to that shown in Fig. 10c.3. The logic high terminating voltage should have a value in between 3.0–3.5 V.

Pull-up (1.5k) and pull-down (15k) resistors are placed at the input or output of a port, shown in Fig. 10c.4, and are inserted for detection of device connections.

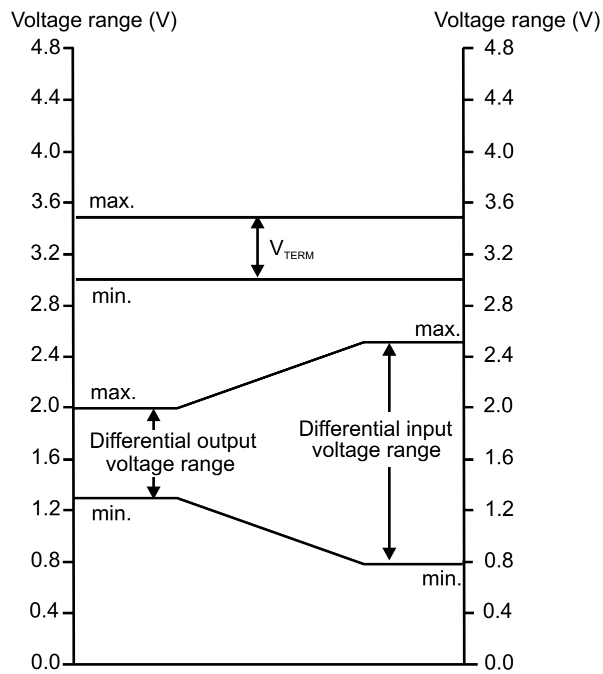


Fig. 10c.3: USB data signal levels

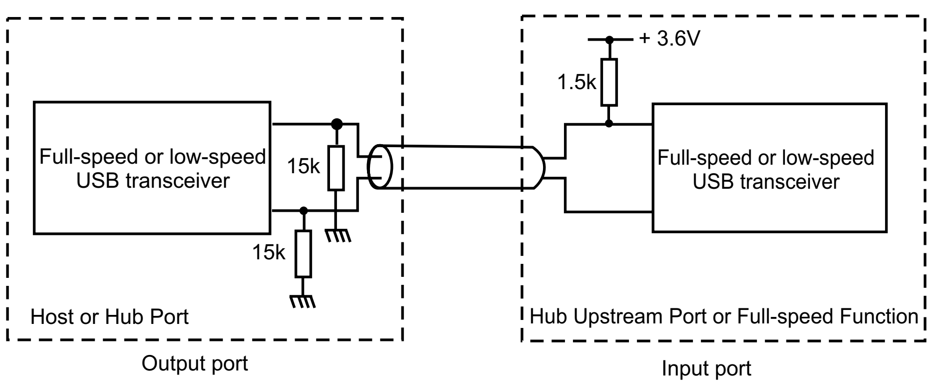


Fig. 10c.4: Pull-up and pull-down resistors in a USB interface

8. Describe the USB Cable.

Ans. The cross-sectional view of the USB cable is shown in Fig. 10c.5.

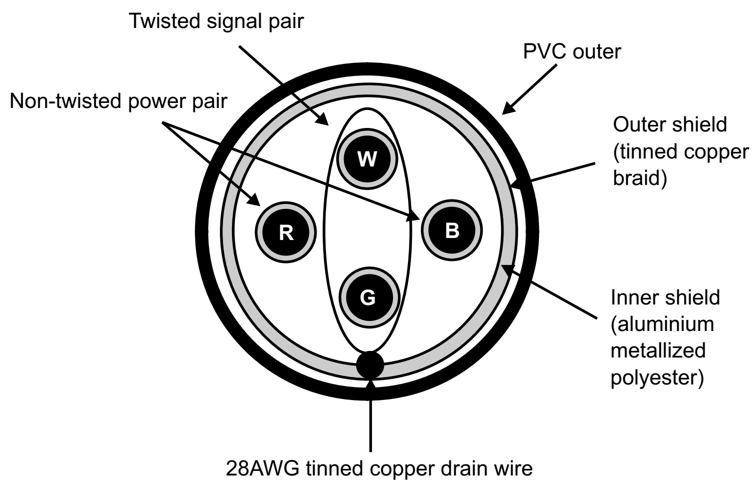


Fig. 10c.5: USB cable (cross-sectional view)

USB cable assemblies are available in three varieties : detachable full-speed captive and low-speed cables. The colours recommended for the cable assembly are white, grey or black.

USB cable consists of four conductors—two for power and two for data. Again two variants are there—full-speed cable and low-speed cable. Full-speed cable consists of twisted pair of conductors for signalling which is not so for low speed cables—because for the latter, noise and electro-magnetic interference are not dominant.

Full-speed cables may be used for low-speed purposes also—but in such a case the full-speed cable must conform to low-speed cable requirements.

9. Discuss the different types of USB connectors.

Ans. There are two types of USB connectors—Series A and Series B or Type A and Type B. 'A' Type connectors are used in USB devices while 'B' type connectors are meant for device vendors in order to provide a standard detachable cable. 'Keyed connector' protocol is used by USB.

Fig. 10c.6 shows a typical type 'A' connector, showing the standard USB icon and a top locator on the cable end side of the connector.

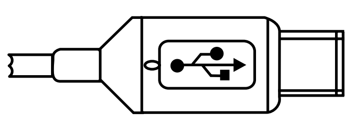


Fig. 10c.6: USB cable connector

Fig. 10c.7 and Fig.10c.8 show respectively Type A and Type B connectors whilst Fig. 10c.9 shows the signals corresponding to the pins and the recommended colour codings also.

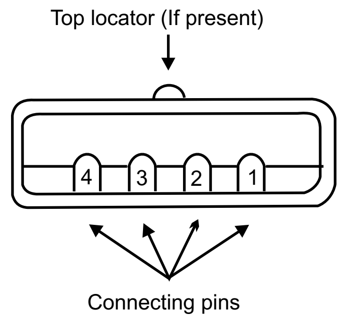


Fig. 10c.7: USB connector (type A)

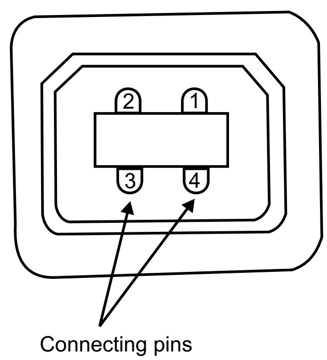


Fig. 10c.8: USB connector (type B)

Pin number	Signal	Recommended colour
1	V_{BUS}	Red
2	D–	White
3	D+	Green
4	GND	Black
Shell	Shield	Drain Wire

Fig. 10c.9: USB pin connections

Full speed devices use B type connector allowing the device to use detachable USB cable. Thus devices can be built without hardwired cable and cable replacement becomes very easy if the need so arises.

The following gives a comparison between Type A and Type B connectors:

Type A connector	**Type B connector**
1. Series A plugs are always oriented towards the host. 2. Series A plug mates with series A receptacle. 3. Series A receptacle acts as outputs from host system/hubs. 4. Series A plugs are oriented towards the host.	1. Series B plugs are always oriented towards the USB device. 2. Series B plug mates with series B receptacle. 3. Series B acts as inputs to hubs/devices. 4. Series B plugs are oriented towards the USB hub/device.

10. Draw and describe the USB topology showing the different layers or tiers.

Ans. A tiered star topology is used to connect USB host and the devices and is shown in Fig. 10c.10. A maximum of 127 USB devices can be connected to one USB bus.

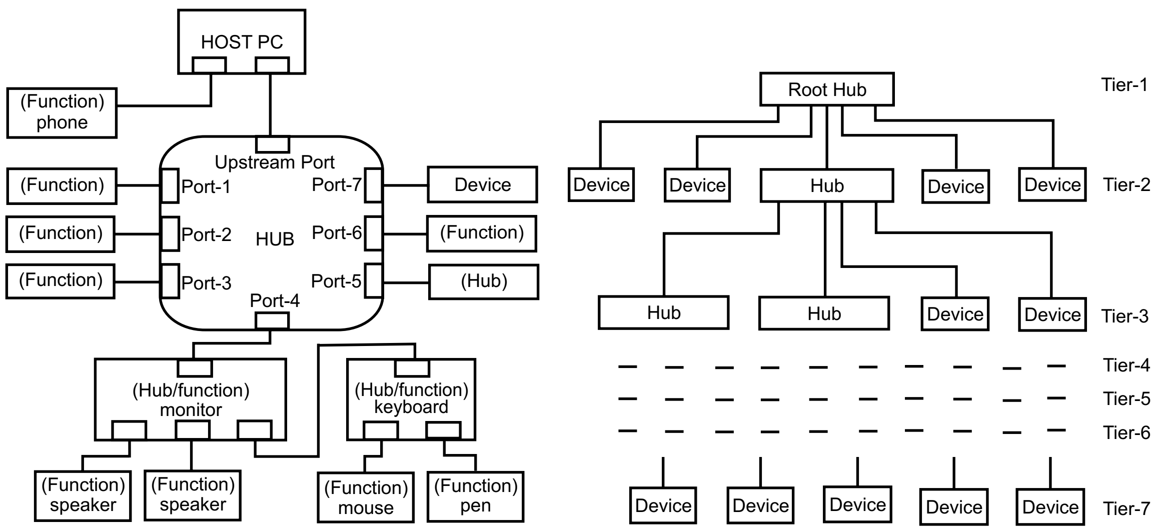


Fig. 10c.10: The USB physical bus topology

11. Discuss bit stuffing and synchronisation field as employed in USB.

Ans. USB devices always employ encoded data for transmission and does not have a separate clock. In such a system, logic 1 is represented by no transition in signal level while logic 0 is represented by a signal transition which is effected at the beginning of the bit interval.

If a given data has long string of 1's then no transition takes place during this period and the receiver may go out of synchronisation. In such a case, on occurrence of six consecutive 1's, the transmitter inserts (or stuffs) a 0. Thus a signal transition is forced to occur after the sixth '1' and enables synchronisation between transmitter and receiver. The receiver, on detecting such a 0, discards it.

But stuffing alone cannot always guarantee synchronisation between transmitter and receiver because the host (PC) and the device do not share the same clock. Each packet transmitted by the transmitter begins with a synchronisation field—it is a byte having the form KJKJKJKK. When the receiver detects such a field, it knows that data packets are going to come down the line. A synchronous field per packet is used.

12. How error detection and its possible elimination is done in USB?

Ans. Differential drivers and receivers are employed and shielding of cables are done to enhance data integrity in USB transmission. Error detection is done by employing CRS (Cyclic Redundancy Character) code on control and data fields. Automatic detection of attachment and detachment of devices and system level configuration of resources are some of the other facilities available in the system. Error protection fields are included in each packet of data transmitted to obviate the effects of transients. An error recovery procedure is invoked in hardware or software for cases where data integrity is of prime importance. The procedure includes retransmission of failed data a fixed number of times, beyond which the software is informed about this repeated failures. The software will then try to retrieve the data—as per the application and device function.

13. Discuss the data transfer types possible in USB.

Ans. Four types of data transfers are allowed by USB architecture. These are: control data transfers, bulk data transfers, interrupt data transfers and isochronous data transfers. These are discussed below:

Control data transfers: It is used by the USB system software when a device is first attached to the system. Apart from this, some other driver software can use the control data transfers in some special cases.

Bulk data transfers: Bulk data transfers are generally meant for printers or scanners, etc.—the data transfer being sequential in nature. Data security is ensured at the hardware level and also invoking retries a limited number of times in case of failures.

Interrupt data transfers: Some event occurrence, characters, coordinates that are organised as groups of one/or a few bytes fall under this category. Such data from a device may occur at any time.

Isochronous data transfers: If data is continuous and delivered in real-time, then it is isochronous data. This kind of data must be delivered the moment it is received.

Voice is an example of isochronous data. Two problems may arise when voice data delivery is undertaken. Assuming data delivered at the appropriate rate by the USB hardware, delivery delays may be there due to software-degrading real-time applications. On the other hand, if delivery rate is not maintained, drop-outs may occur due to buffers or frame underruns or overruns. To ensure proper data delivery, sometimes a part of the available USB bandwidth is reserved/dedicated for isochronous data transfers.

14. Describe the clock signal associated with USB.

Ans. The clock signal associated with USB is encoded. The form of encoding undertaken is NRZI with bit stuffing. This ensures adequate transitions. Before sending each packet a SYNC field is sent. This is done for the receiver to get itself ready to synchronise itself with the bit recovery clock.

15. Describe how data is transported between host controller and devices via USB.

Ans. It is the host controller (i.e., the PC) which initiates all the data transfers on USB. Up to three packets are transmitted for each bus transaction. The host controller sends a USB packet that contains the type and direction transaction, the device address on the USB and the endpoint number. This first packet is called a token packet and a must whenever a transaction is initiated.

The device address which is received by each of the devices on the USB are decoded and the one whose address matches with the address which has been sent (via the token packet) is identified on the USB. The token packet also contains the information about the directions of data flow—i.e., whether from a host to device or vice-versa. The source

then sends the data packet. The destination responds with a handshake packet indicating whether the transfer was successful.

16. What are the different devices available as far as device powering is concerned?

Ans. There are two types of devices—bus-powered devices and self-powered devices.

Devices which depend totally on power from the cable are called bus-powered devices while devices whose power are made available otherwise are called self-powered devices. Devices are powered by the connected hubs and this power may be available from the host controller or from an external power source.

Cables and connectors are such that upstream and downstream connectors are not interchangeable—thus loopback connections at the hubs are not feasible.

17. Show the different layers in USB architecture and discuss.

Ans. The different layers in the USB architecture are shown in Fig. 10c.11. The different layers shown in the architecture are Bus Interface Layer, Device Layer and Function Layer. The Bus Interface Layer provides physical/signalling/packet connectivity between the host and a device. This layer is the one which undertakes the actual data transfer. Each layer provides peer-to-peer connectivity. The Device Layer establishes logical connectivity between operating system software and USB logical device whereas the Function layer provides logical interconnection between the application software and USB functions.

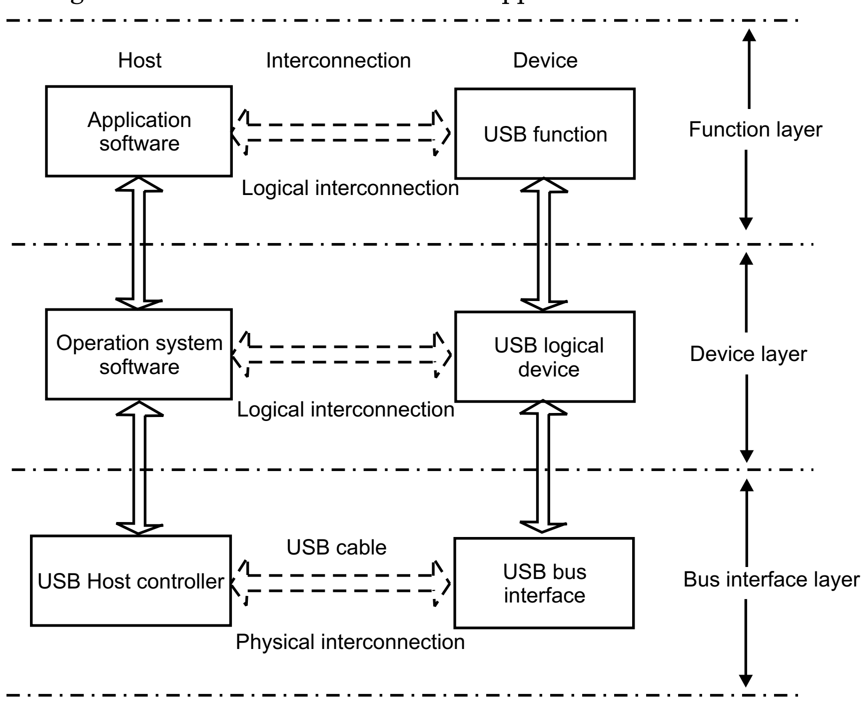


Fig. 10c.11: Layers in USB architecture (note the apparent peer-peer logic interconnection)

18. Discuss connection/disconnection of USB devices from the bus.

Ans. Host-connection and host-disconnection of devices from the USB is possible—this is a very vital advantage with USB which is not available with other bus systems. The host's (PC's)

software is thus able to recognise the connection/disconnection while it is in service and is able to reconfigure the system dynamically.

Status indicators are available on particular pins of a port attached to a hub that indicate the connection/disconnection of a USB device. The host accesses these pins to get to know the status of a device.

A device, if newly attached, is assigned a particular address by the host and it (the host) determines via software whether the newly attached device is a hub or a function.

When a device is detached from a hub's port, the hub disables the port and indicate the host about the removal of the device. On the other hand, if the detached device is a hub the system software handles this removal of the hub(s) and or device which are connected to the removed hub.

19. What is enumeration?

Ans. The process of allocating a unique address to a unique device is called enumeration. Since attachment/detachment of devices to USB can take place at any time, hence enumeration is an on-going process as far as the USB system software is concerned.

20. What is meant by Endpoint?

Ans. Information regarding configuration, set-up or data are transported on the USB via buffers of definite size. These buffers are called endpoints. Two types of endpoints are there —IN endpoints and OUT endpoints. The host in the system delivers information into an OUT endpoint (towards the device) and the device delivers information into an IN endpoint (towards the host). An USB device can have a maximum of 16 IN and 16 OUT endpoints.

Again endpoints are of two types—data endpoint and control endpoint. Each endpoint has an unique endpoint number.

21. Discuss packets with regard to data transfer.

Ans. Information is transferred in packets in USB. Four packet types are used—token, data, handshake and start of frame packets. Each type of packet contains a combination of the following fields:

Synchronization	(SYNC)
Packet ID	(PID)
Address	(ADDR)
Endpoint	(ENDP)
Error checking	(CRC) and
End of packet	(EOP)

SYNC, PID and EOP are common in all the four types of packets.

SYNC is a 1-byte sequence KJKJKJKK. All packets start with SYNC field. PID is also a one byte field that identifies the type of packet being sent. The lower nibble indicates the PLD, while the upper nibble is a complement of the lower one. ADDR is a 7-bit field and contains the address of a function of the USB device for which the packet is meant. On resetting or powering on of a device, the default address becomes 0. It is reprogrammed during enumeration by the host. ENDP is a one nibble field and hence a maximum of 16 endpoints can be addressed. CRC is 5-bit/16-bit for token/data packets respectively. EOP contains two bits in SEO state and is followed by a single bit in J state.

22. Show the formats of the four types of packets and discuss them.

Ans. The four types of packets are shown below in Fig. 10c.12.

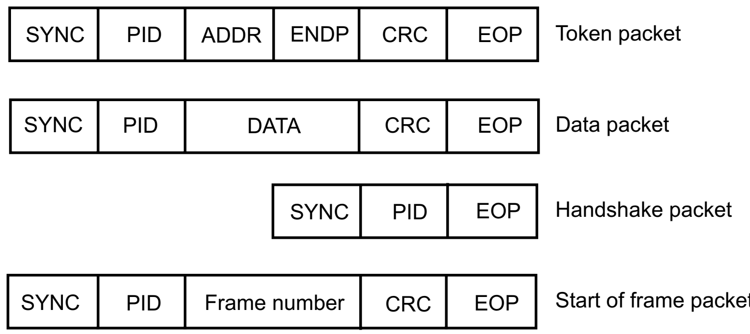


Fig.10c.12: Formats of token, data, handshake and start of frame packets

Token Packet: Token packet is a must for each transaction and is sent by the host controller at the beginning of each transaction. This packet indicates the type of transaction that is to take place. The ADDR (address field) is decoded by the USB device and it selects itself if the address is meant for it. Token packets can be of three types : IN, OUT and SETUP. The first indicates that the host intends to read information while the second indicates that the host would send information. SETUP indicates beginning of transfer of control.

Data Packet: A data packet may contain between 0 to 1023 bytes of data. Four types of data packets are there : DATA0, DATA1, DATA2 and MDATA. Data transfer takes place either way from host to device or vice-versa i.e., downstream or upstream.

Handshake Packet: It is used for successful reception of data, flow control, halt condition, command acceptance/rejection, etc. It can be of three types : ACK, NAK and STALL. ACK indicates that the packet has been successfully received while NAK (negative acknowledgement) indicates data packet not received properly or else the receiver is busy. STALL indicates a function is unable to transmit/receive data.

Start-of-Frame Packet: It is issued by the host at the beginning of each frame. Frame transfer period is 125 μs for high speed and 1 ms for low speed or full speed bus type. The host increments the frame number each time a frame gets transferred. After reaching a value of 7FF H, the frame number rolls over.

23. Describe USB descriptors.

Ans. A hierarchy of descriptors are there for USB devices. These descriptors are data structures. They carry all the informations about a device, how the device is configured, the number as also the types of endpoints a particular device has etc.

The different descriptors are:

- Device descriptors
- Interface descriptors
- Endpoint descriptors
- Configuration descriptors
- String descriptors etc.

To note at this point that all the descriptors have a common format.

11

The 8086 Microprocessor

1. Draw the pin diagram of 8086.

Ans. There would be two pin diagrams—one for MIN mode and the other for MAX mode of 8086, shown in Figs. 11.1 and 11.2 respectively. The pins that differ with each other in the two modes are from pin-24 to pin-31 (total 8 pins).

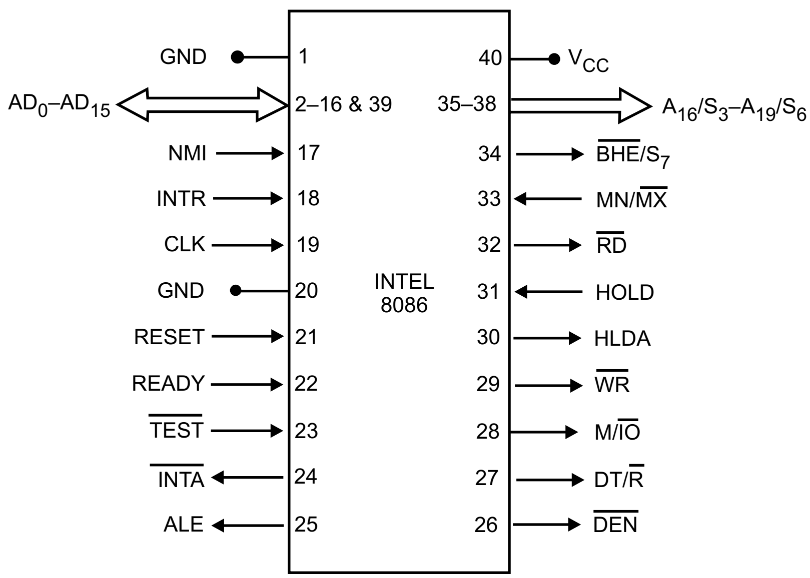


Fig. 11.1: Signals of intel 8086 for minimum mode of operation

2. What is the technology used in 8086 μP?

Ans. It is manufactured using high performance metal-oxide semiconductor (HMOS) technology. It has approximately 29,000 transistors and housed in a 40-pin DIP package.

3. Mention and explain the modes in which 8086 can operate.

Ans. 8086 μP can operate in two modes—MIN mode and MAX mode.

When $MN/\overline{MX}$ pin is high, it operates in MIN mode and when low, 8086 operates in MAX mode.

For a small system in which only one 8086 microprocessor is employed as a CPU, the system operates in MIN mode (Uniprocessor). While if more than one 8086 operate in a system then it is said to operate in MAX mode (Multiprocessor).

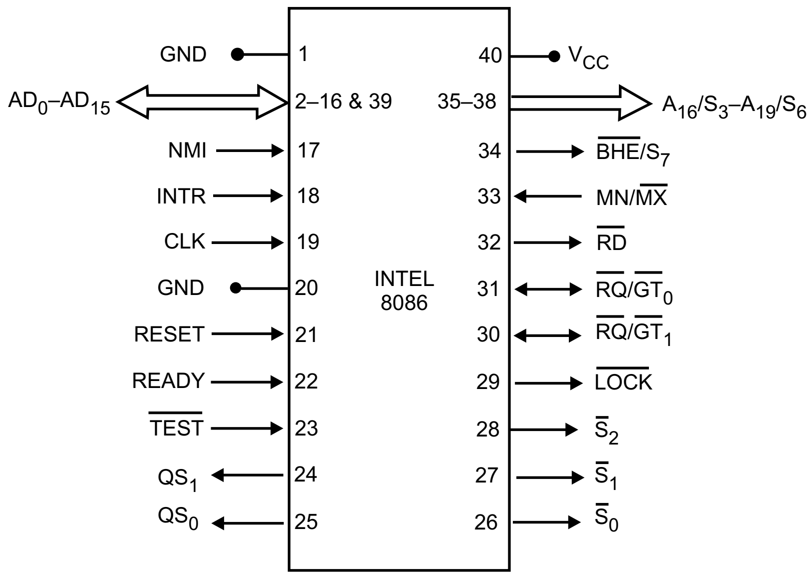


Fig. 11.2: Signals of intel 8086 for maximum mode of operation

The bus controller IC (8288) generates the control signals in case of MAX mode, while in MIN mode CPU issues the control signals required by memory and I/O devices.

4. Distinguish between the lower sixteen address lines from the upper four.

Ans. Both the lower sixteen address lines $(AD_0 - AD_{15})$ and the upper four address lines $(A_{16}/S_3 - A_{19}/S_6)$ are multiplexed.

During T_1, the lower sixteen lines carry address $(A_0 - A_{15})$, while during T_2, T_3 and T_4 they carry data.

Similarly during T_1, the upper four lines carry address (A_{16}– A_{19}), while during T_2, T_3 and T_4, they carry status signals.

5. In how many modes the minimum-mode signal can be divided?

Ans. In the MIN mode, the signals can be divided into the following basic groups: address/data bus, status, control, interrupt and DMA.

6. Tabulate the common signals, Minimum mode signals and Maximum mode signals. Also mention their functions and types.

Ans. Table 11.1 shows the common signals, Minimum mode signals and the Maximum mode signals, along with the functions of each and their types.

Table 11.1 : (a) Signals common to both minimum and maximum mode, (b) Unique minimum-mode signals, (c) Unique maximum-mode signals for 8086.

Common signals		
Name	**Function**	**Type**
AD15-AD0	Address/data bus	Bidirectional, 3–state
A19/S6-A16/S3	Address/status	Output, 3-state
MN/$\overline{\text{MX}}$	Minimum/maximum Mode control	Input
$\overline{\text{RD}}$	Read control	Output, 3-state
$\overline{\text{TEST}}$	Wait on test control	Input
READY	Wait state control	Input
RESET	System reset	Input
NMI	Nonmasakable Interrupt request	Input
INTR	Interrupt request	Input
CLK	System clock	Input
V_{cc}	+5V	Input
GND	Ground	

(a)

Minimum mode signals (MN/$\overline{\text{MX}}$ =V_{cc})		
Name	**Function**	**Type**
HOLD	Hold request	Input
HLDA	Hold acknowledge	Output
$\overline{\text{WR}}$	Write control	Output, 3-state
M/$\overline{\text{IO}}$	IO/memory control	Output, 3-state
DT/$\overline{\text{R}}$	Data transmit/receive	Output, 3-state
$\overline{\text{DEN}}$	Data enable	Output, 3-state
ALE	Address latch enable	Output
$\overline{\text{INTA}}$	Interrupt acknowledge	Output

(b)

Maximum mode signals (MN/$\overline{\text{MX}}$ =GND)		
Name	**Function**	**Type**
$\overline{\text{RQ}}/\overline{\text{GT1,0}}$	Request/grant bus access control	Bidirectional
$\overline{\text{LOCK}}$	Bus priority lock control	Output, 3-state
$\overline{\text{S2}}-\overline{\text{S0}}$	Bus cycle status	Output, 3-state
QS1, QS0	Instruction queue status	Output

(c)

7. Mention the different varieties of 8086 and their corresponding speeds.

Ans. The following shows the different varieties of 8086 available and their corresponding speeds.

Types	Speeds
8086	5 MHz
8086–1	10 MHz
8086–2	8 MHz

8. Mention (a) the address capability of 8086 and (b) how many I/O lines can be accessed by 8086.

Ans. 8086 addresses via its A_0–A_{19} address lines. Hence it can address 2^{20} = 1MB memory.

Address lines A_0 to A_{15} are used for accessing I/O's. Thus, 8086 can access 2^{16} = 64 KB of I/O's.

9. What is meant by microarchitecture of 8086?

Ans. The individual building blocks of 8086 that, as a whole, implement the software and hardware architecture of 8086. Because of incorporation of additional features being necessitated by higher performance, the microarchitecture of 8086 or for that matter any microprocessor family, evolves over time.

10. Draw and discuss the architecture of 8086. Mention the jobs performed by BIU and EU.

Ans. The architecture of 8086 is shown below in Fig. 11.3. It has got two separate functional units—Bus Interface Unit (BIU) and Execution Unit (EU).

8086 architecture employs parallel processing—i.e., both the units (BIU and EU) work at the same time. This is *Unlike* 8085 in which *Sequential* fetch and execute operations take place. Thus in case of 8086, efficient use of system bus takes place and higher performance (because of reduced instruction time) is ensured.

- BIU has segment registers, instruction pointer, address generation and bus control logic block, instruction queue while the EU has general purpose registers, ALU, control unit, instruction register, flag (or status) register.

The main jobs performed by BIU are:

- BIU is the 8086's interface to the outside world, i.e., all *External* bus operations are done by BIU.
- It does the job of instruction fetching, reading/writing of data/operands for memory and also the inputting/outputting of data for peripheral devices.
- It does the job of filling the instruction queue.
- Does the job of address generation.

The main jobs performed by the execution unit are:

- Decoding/execution of instructions.
- It accepts instructions from the output end of instruction queue (residing in BIU) and data from the general purpose registers or memory.
- It generates operand addresses when necessary, hands them over to BIU requesting it (BIU) to perform read or write cycle to memory or I/O devices.
- EU tests the status of flags in the control register and updates them when executing instructions.
- EU waits for instructions from the instruction queue, when it is empty.
- EU has no connection to the system buses.

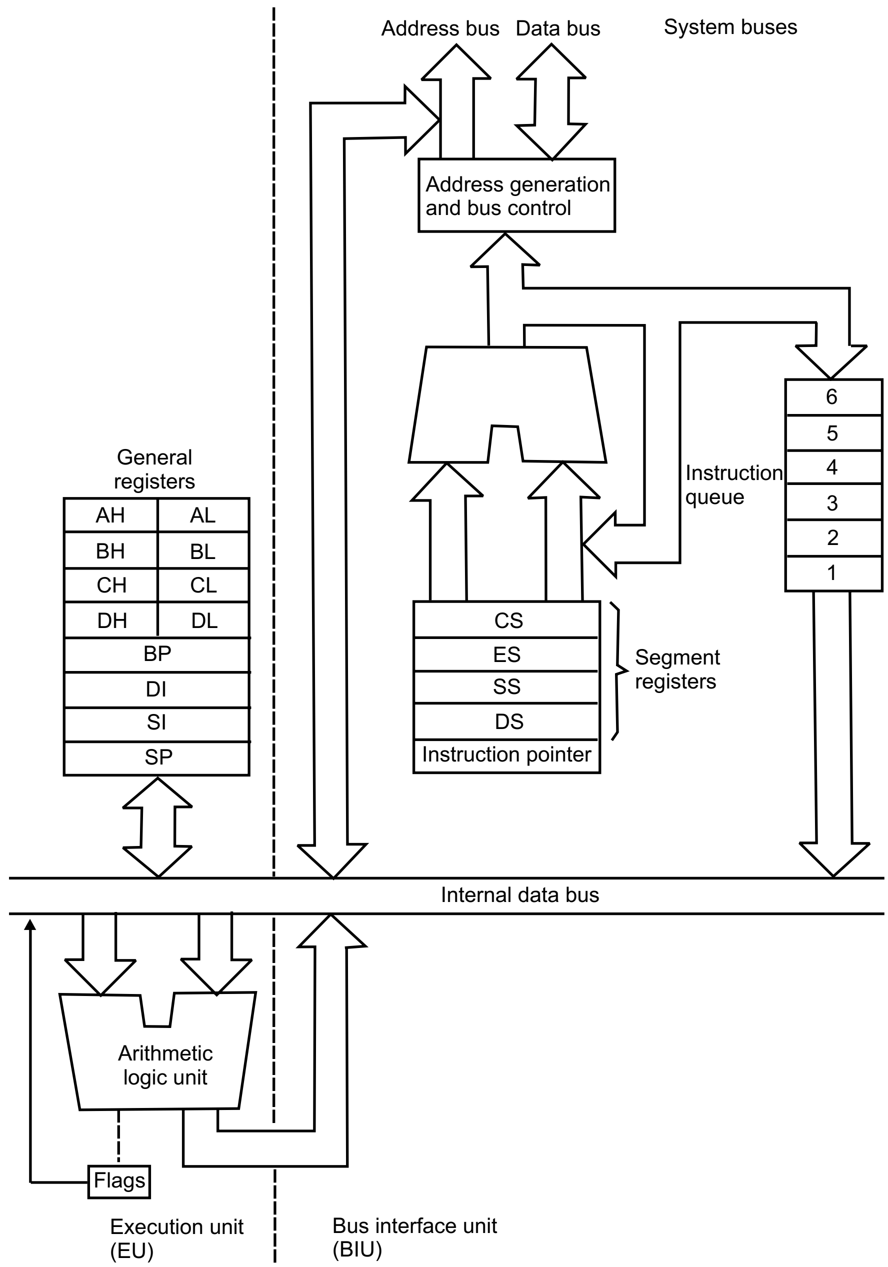


Fig. 11.3: CPU model for the 8086 microprocessor. A separate execution unit (EU) and bus interface unit (BIU) are provided.

11. Explain the operations of instructions queue residing in BIU.

Ans. The instruction queue is 6-bytes in length, operates on FIFO basis, and receives the instruction codes from memory. BIU fetches the instructions meant for the queue ahead of time from memory. In case of JUMP and CALL instructions, the queue is dumped and newly formed from the new address.

Because of the instruction queue, there is an overlap between the instruction execution and instruction fetching. This feature of fetching the next instruction when the current instruction is being executed, is called *Pipelining*.

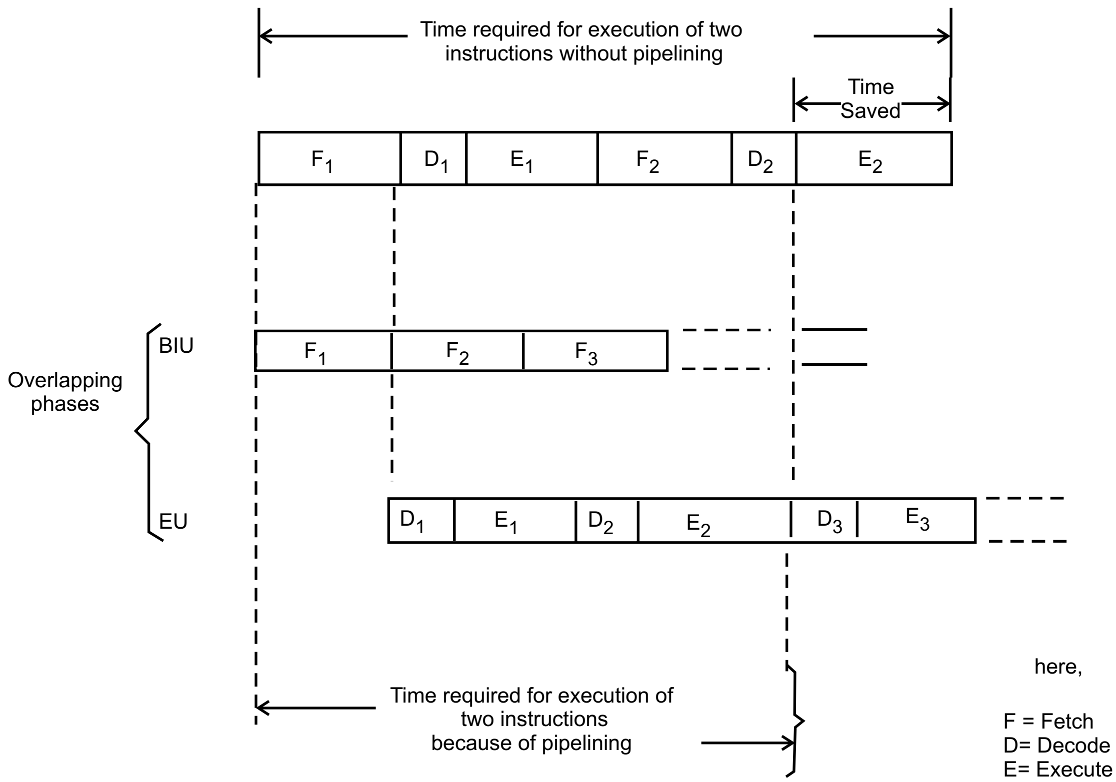


Fig.11.4: Pipelining procedure saves time

Fig. 11.4, which is self-explanatory, shows that there is definitely a time saved in case of overlapping phases (as in the case of 8086) compared to sequential phases (as in the case of 8085).

Initially, the queue is empty and CS : IP is loaded with the required address (from which the execution is to be started). Microprocessor 8086 starts operation by fetching 1 (or 2) byte(s) of instruction code(s) if CS : IP address is odd (even).

The 1st byte is always an opcode, which when decoded, one byte in the queue becomes empty and the queue is updated. The filling in operation of the queue is not started until two bytes of the instruction queue is empty. The instruction execution cycle is *never* broken for fetch operation.

After decoding of the 1st byte, the decoder circuit gets to know whether the instruction is of single or double opcode byte.

For a single opcode byte, the next bytes are treated as data bytes depending upon the decoded instruction length, otherwise the next byte is treated as the second byte of the instruction opcode.

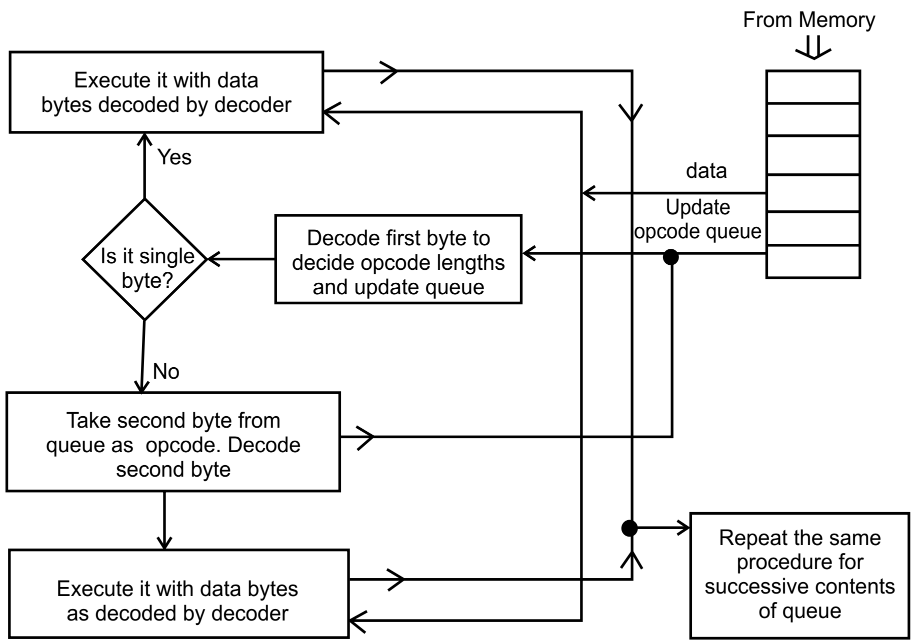


Fig.11.5 : The queue operation

For a 2-byte instruction code, the decoding process takes place taking both the bytes into consideration which then decides on the decoded instruction length and the number of subsequent bytes which will be treated as instruction data. Updation of the queue takes place once a byte is read from the queue.

The queue operation is shown in Fig. 11.5 in block schematic form.

12. Mention the conditions for which EU enters into WAIT mode.

Ans. There are three conditions that cause the EU to enter into WAIT state. These are:

- When an instruction requires the access to a memory location not in the queue.
- When a JUMP instruction is executed. In this case the current queue contents are aborted and the EU waits until the instructions at the jump address is fetched from memory.
- During execution of instruction which are very slow to execute. The instruction AAM (ASCII adjust for multiplication) requires 83 clock cycles for execution. For such a case, the BIU is made to wait till EU pulls one or two bytes from the queue before resuming the fetch cycle.

13. Mention the kind of operations possible with 8086.

Ans. It can perform bit, byte, word and block operations. Also multiplication and division operations can be performed by 8086.

14. Mention the total number of registers of 8086 and show the manner in which they are grouped.

Ans. There are in all fourteen numbers of 16-bit registers. The different groups are made as hereunder:

- Data group, pointers and index group, status and control flag group and segment group.
- The data group consists of AX (accumulator), BX (base), CX (count) and DX (data).
- Pointer and Index group consist of SP (Stack pointer), BP (Base pointer), SI (Source Index), DI (Destination index) and IP (Instruction pointer).
- Segment group consists of ES (Extra Segment), CS (Code Segment), DS (Data Segment) and SS (Stack Segment).
- Control flag group consists of a single 16-bit flag register.

Fig. 11.6 shows the registers placed in the different groups to form a programming model.

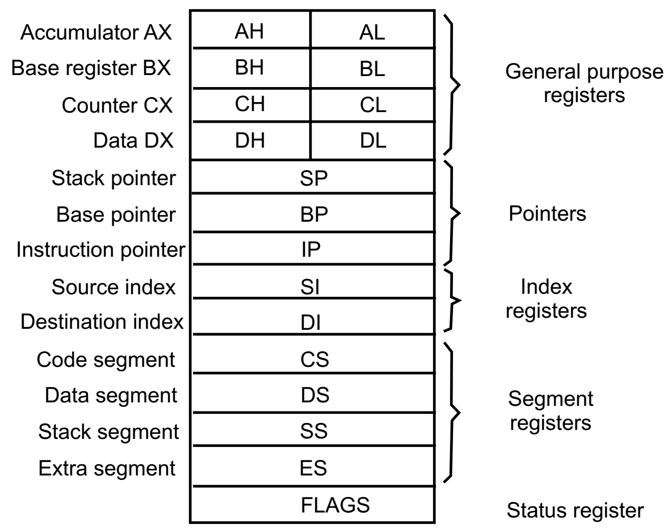
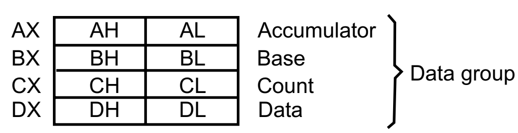


Fig.11.6: Schematic diagram of intel 8086 registers

15. Describe, in detail, the general purpose of data registers.

Ans. Fig. 11.7 shows the four data registers along with their dedicated functions also.



Register	Operations
AX	Word multiply, Word divide, Word I/O
AL	Byte multiply, byte divide, byte I/O, translate, decimal arithmetic
AH	Byte multiply, byte divide
BX	Translate
CX	String operations, Loops
CL	Variable shift and rotate
DX	Word multiply, Word divide, indirect I/O

Fig. 11.7: Data group registers and their functions

All the four registers can be used bytewise or wordwise. The alphabets X, H and L respectively refer to word, higher byte or lower byte respectively of any register.

All the four registers can be used as the source or destination of an operand during an arithmetic operation such as ADD or logical operation such as AND, although particular registers are earmarked for specific operations. Register C is used as a count register in string operations and as such is called a 'count' register. Register C is also used for multibit shift or rotate instructions.

Register D is used to hold the address of I/O port while register A is used for all I/O operations that require data to be inputted or outputted.

16. Describe the status register of 8086.

Ans. It is a 16-bit register, also called flag register or Program Status Word (PSW). Seven bits remain unused while the rest nine are used to indicate the conditions of flags. The status flags of the register are shown below in Fig. 11.8.

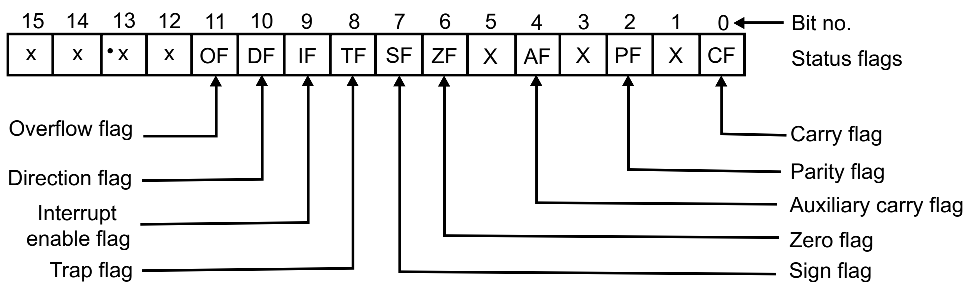


Fig.11.8: Status flags of intel 8086

Out of nine flags, six are condition flags and three are control flags. The control flags are TF (Trap), IF (Interrupt) and DF (Direction) flags, which can be set/reset by the programmer, while the condition flags [OF (Overflow), SF (Sign), ZF (Zero), AF (Auxiliary Carry), PF (Parity) and CF (Carry)] are set/reset depending on the results of some arithmetic or logical operations during program execution.

CF is set if there is a carry out of the MSB position resulting from an addition operation or if a borrow is needed out of the MSB position during subtraction.

PF is set if the lower 8-bits of the result of an operation contains an even number of 1's. AF is set if there is a carry out of bit 3 resulting from an addition operation or a borrow required from bit 4 into bit 3 during subtraction operation.

ZF is set if the result of an arithmetic or logical operation is zero.

SF is set if the MSB of the result of an operation is 1. SF is used with unsigned numbers.

OF is used only for signed arithmetic operation and is set if the result is too large to be fitted in the number of bits available to accommodate it.

The functions of the flags along with their bit positions are shown in Fig. 11.9 below.

Bit position	Name	Function
0	CF	Carry flag: Set on high-order bit carry or borrow; cleared otherwise
2	PF	Parity flag: Set if low-order 8-bit of result contain an even number of 1-bit; cleared otherwise
4	AF	Set on carry from or borrow to the low-order 4-bits of AL; cleared otherwise

6	ZF	Zero flag: Set if result is zero; cleared otherwise
7	SF	Sign Flag: Set equal to high-order bit of result (0 is positive, 1 if negative)
8	TF	Signal step flag: Once set, a single-step interrupt occurs after the next instruction executes; TF is cleared by the single-step interrupt
9	IF	Interrupt-enable flag: When set, maskable interrupts will cause the CPU to transfer control to an interrupt vector specified location.
10	DF	Direction flag: Causes string instructions to auto decrement the appropriate index register when set; clearing DF causes auto increment.
11	OF	Overflow flag: Set if the signed result cannot be expressed within the number of bits in the destination operand; cleared otherwise.

Fig.11.9: 8086 flags: DF, IF and TF can be set or reset to control the operations of the processor. The remaining flags are status indicators.

17. Discuss the three control flags of 8086.

Ans. The three control flags of 8086 are TF, IF and DF. These three flags are programmable, i.e., can be set/reset by the programmer so as to control the operation of the processor.

When TF (trap flag) is set (=1), the processor operates in single stepping mode—i.e., pausing after each instruction is executed. This mode is very useful during program development or program debugging.

When an interrupt is recognised, TF flag is cleared. When the CPU returns to the main program from ISS (interrupt service subroutine), by execution of IRET in the last line of ISS, TF flag is restored to its value that it had before interruption.

TF cannot be directly set or reset. So indirectly it is done by pushing the flag register on the stack, changing TF as desired and then popping the flag register from the stack.

When IF (interrupt flag) is set, the maskable interrupt INTR is enabled otherwise disabled (i.e., when IF = 0).

IF can be set by executing STI instruction and cleared by CLI instruction. Like TF flag, when an interrupt is recognised, IF flag is cleared, so that INTR is disabled. In the last line of ISS when IRET is encountered, IF is restored to its original value.

When 8086 is reset, IF is cleared, i.e., resetted.

DF (direction flag) is used in string (also known as block move) operations. It can be set by STD instruction and cleared by CLD. If DF is set to 1 and MOVS instruction is executed, the contents of the index registers DI and SI are automatically decremented to access the string from the highest memory location down to the lowest memory location.

18. Discuss the Pointers and Index group of registers.

Ans. The pointer registers are SP and BP while the index registers are SI and DI.

All the four are 16-bit registers and are used to store offset addresses of memory locations relative to segment registers. They act as memory pointers. As an example, MOV AH, [SI] implies, "Move the byte whose address is contained in SI into AH". If now, SI = 2000 H, then execution of above instruction will put the value FF H in register AH, shown in Fig. 11.10, [SI+1 : SI] = ABFF H, where obviously SI+1 points to memory location 2001 H and [SI+1] = AB H.

SI and DI are also used as general purpose registers. Again in certain string (block move) instructions, SI and DI are used as source and destination index registers

respectively. For such cases, contents of SI are added to contents of DS register to get the actual source address of data, while the contents of DI are added to the contents of ES to get the actual destination address of data.

SP and BP stand for stack pointer and base pointer with SP containing the offset address or the stack top address. The actual stack address is computed by adding the contents of SP and SS.

Data area(s) may exist in stack. To access such data area in stack segment, BP register is used which contains the offset address. BP register is also used as a general purpose register.

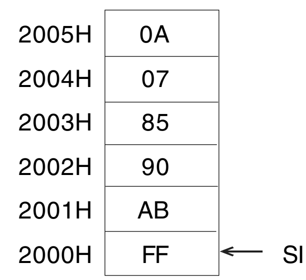


Fig. 11.10: SI is pointing to memory locations 2000 H.

Instruction pointer (IP) is also included in the index and pointers group. IP points to the offset instead of the actual address of the next instruction to be fetched (from the current code segment) in BIU. IP resides in BIU but cannot be programmed by the programmer.

19. Describe in brief the four segment registers.

Ans. The four segment registers are CS, DS, ES and SS—standing for code segment register, data segment register, extra segment register and stack segment register respectively. When a particular memory is being read or written into, the corresponding memory address is determined by the content of one of these four segment registers in conjunction with their offset addresses.

The contents of these registers can be changed so that the program may jump from one active code segment to another one.

The use of these segment registers will be more apparent in memory segmentation schemes.

20. Discuss A_{16}/S_3—A_{19}/S_6 Signals of 8086.

Ans. These are time multiplexed signals. During T_1, they represent $A_{19} – A_{16}$ address lines. During I/O operations, these lines remain low. During T_2–T_4, they carry status signals.

S_4 and S_3 (during T_2 to T_4) identify the segment register employed for 20-bit physical address generation.

Status signal S_5 (during T_2 to T_4) represents interrupt enable status. This is updated at the beginning of each clock cycle.

Status signal S_6 remains low during T_2 to T_4.

21. Discuss $\overline{BHE}/S_7$ signal.

Ans. During T_1, this becomes bus high enable signal and remains low while during T_2 to T_4 it acts as a status signal S_7 and remains high during this time.

During T_1, when $\overline{BHE}$ signal is active, i.e., remains low, it is used as a chip select signal on the higher byte of data bus—i.e., D_{15}–D_8.

Table 11.2 shows $\overline{BHE}$ and A_0 signals determine one of the three possible references to memory.

Table 11.2: Status of $\overline{BHE}$ and A_0 identify memory references

$\overline{BHE}$	A_0	Word/byte access
0	0	Both banks active, 16-bit word transfer on $AD_{15} – AD_0$
0	1	Only high bank active, upper byte from/to odd address on $AD_{15} – AD_8$
1	0	Only low bank active, lower byte from/to even address $AD_7–AD_0$
1	1	No bank active

22. Discuss the Reset pin of 8086.

Ans. Reset is an active high input signal and must be active for at least 4 CLK cycles to be accepted by 8086. This signal is internally synchronised and execution starts only after Reset returns to low value.

For proper initialisation, Reset pulse must *not* be applied before 50µS of 'power on' of the circuit. During Reset state, all three buses are tristated and ALE and HLDA are driven low.

During resetting, all internal register contents are set to 0000 H, but CS is set to F000 H and IP to FFF0 H. Thus execution starts from physical address FFFF0 H. Thus EPROM in 8086 is interfaced so as to have the physical memory location forms FFFF0 H to FFFFF H, i.e., at the end of the map.

23. Discuss the two pins (a) DT/$\overline{R}$ and (b) $\overline{DEN}$.

Ans. (a) DT/$\overline{R}$ is an output pin which decides the directions of data flow through the transreceivers (bidirectional buffers).

When the processor sends out data, this signal is 1 while when it receives data, the signal status is 0.

(b) $\overline{DEN}$ stands for data enable. It is an active low signal and indicates the availability of data over the address/data lines. This signal enables the transreceivers to separate data from the multiplexed address/data signal. It is active from the middle of T_2 until the middle of T_4.

Both DT/$\overline{R}$ and $\overline{DEN}$ are tristated during 'hold acknowledge'.

24. Elaborate the functions of the pins $\overline{S_2}$, $\overline{S_1}$ and $\overline{S_0}$.

Ans. These three are output status signals in the MAX mode, indicating the type of operation carried out by the processor.

The signals become active during T_4 of the previous cycle and remain active during T_1 and T_2 of the current cycle. They return to the passive state during T_3 of the current bus cycle so that they may again become active for the next bus cycle during T_4. Table 11.3 shows the different bus cycles of 8086 for different combinations of these three signals.

Table 11.3: Bus status codes

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	CPU cycles
0	0	0	Interrupt acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	HALT
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive

25. Explain the $\overline{\text{LOCK}}$ signal.

Ans. It is an active low output signal and is activated by LOCK prefix instruction and remains active until the completion of the next instruction. It floats to tri-state during hold acknowledge when $\overline{\text{LOCK}}$ signal is low, all interrupts get masked and HOLD request is not granted. $\overline{\text{LOCK}}$ signal is used by the processor to prevent other devices from accessing the system control bus. This symbol is used when CPU is executing some critical instructions and through this signal other devices are informed that they should not issue HOLD signal to 8086.

26. Explain the $\overline{\text{TEST}}$ signal.

Ans. It is an active low input signal. Normally the BUSY pin (output) of 8087 NDP is connected to the $\overline{\text{TEST}}$ input pin of 8086. When maths co-processor 8087 is busy executing some instructions, it pulls its BUSY signal high. Thus the $\overline{\text{TEST}}$ signal of 8086 is consequently high, and it (8086) is made to WAIT until the BUSY signal goes low. When 8087 completes its instruction executions, BUSY signal becomes low. Thus the $\overline{\text{TEST}}$ input of 8086 becomes low also and then only 8086 goes in for execution of its program.

27. Show how demultiplexing of address/data bus is done and also show the availability of address/data during read/write cycles.

Ans. The demultiplexing of lower 2-bytes of address/data bus (AD_0–AD_{15}) is done by 8282/8283 octal latch with 8282 providing non-inverting outputs while 8283 gives out inverted outputs. The chip outputs are also buffered so that more drive is available at their outputs.

A D latch is central to the demultiplexing operations of these latches. During T_1 when ALE is high, the latch is transparent and the output of latch is 'A' (address) only. At the end of T_1, ALE has a high to low transition which latches the address available at the D input of the latch, so that address is continued to be available from the Q output of the latch (i.e., whole of T_1 to T_4 states).

It is to be noted that memory and I/O devices do not access the data bus until the beginning of T_2, thus the 'data' is a 'don't care' till the end of T_1. This is shown in Fig. 11.11 and the timing diagram shows the availability of data for read and write cycles.

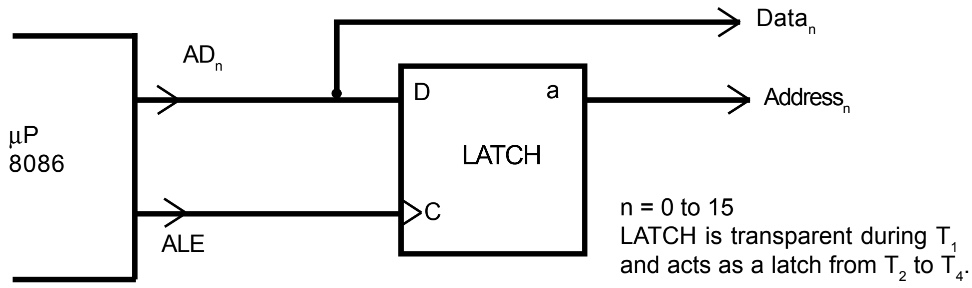


Fig.11.11: Demultiplexing the 8086 address/data bus

28. Discuss the Instruction Pointer (IP) of 8086.

Ans. Functionally, IP plays the part of Program Counter (PC) in 8085. But the difference is that IP holds the offset of the next word of the instruction code instead of the actual address (as in PC).

IP along with CS (code segment) register content provide the 20-bit physical (or real) address needed to access the memory. Thus CS:IP denotes the value of the memory address of the next code (to be fetched from memory).

Content of IP gets incremented by 2 because each time a word of code is fetched from memory.

29. Indicate the data types that can be handled by 8086 μP.

Ans. The types of data formats that can be handled by 8086 fall under the following categories:

- Unsigned or signed integer numbers—both byte-wide and word-wide.
- BCD numbers—both in packed or unpacked form.
- ASCII coded data. ASCII numbers are stored one number per byte.

30. Compare 8086 and 8088 microprocessors.

Ans. The Comparison between the two is tabulated below in Table 11.4.

Table 11.4: Comparison of 8086 and 8088

8086	8088
1. 2-byte data width, obtained by demultiplexing AD_0–AD_{15}.	1. 1-byte data width, obtained by demultiplexing AD_0–AD_7.
2. In MIN mode, pin-28 is assigned the signal $M/\overline{IO}$.	2. In MIN mode, pin-28 is assigned the signal $IO/\overline{M}$.
3. A 6-byte instruction queue.	3. A 4-byte instruction queue.
4. To access higher byte, $\overline{BHE}$ signal is used.	4. No such signal required, since data width is 1-byte only.
5. BIU dissimilar, but EU similar to 8088. Program instructions identical to 8088.	5. BIU dissimilar, but EU similar to 8086. Program instructions identical to 8086.
6. Program fetching from memory done only when 2-bytes are empty in queue.	6. Program fetching from memory done as soon as a byte is free in queue.
7. Pin-34 is $\overline{BHE}/S_7$. During T_1, $\overline{BHE}$ is used to enable data on D_8–D_{15}. During T_2–T_4, status of this pin is 0. In MAX mode, 8087 monitors this pin to identify the CPU—8086 or 8088? Accordingly it sets its queue length to 6 or 4 respectively.	7. Pin-34 is $\overline{SS_0}$. It acts as $\overline{S_0}$ in the MIN mode. In MAX mode $\overline{SS_0}$ = 1 always.

31. Comment on the instruction size of 8086.

Ans. It varies from 1 to 6 bytes.

32. Discuss the instruction format of 8086.

Ans. The instruction format of 8086 is shown in Fig. 11.12. It is extendable up to 6-bytes. The first byte contains D and W—Direction Register Bit and Data Size Bit respectively. Both D and W are 1-bit in nature.

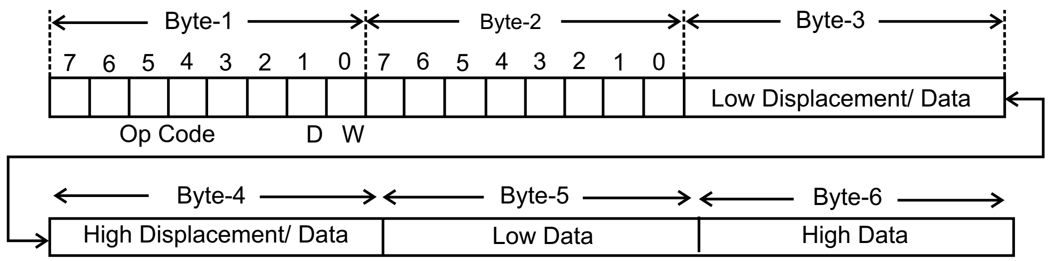


Fig. 11.12: 8086 Instruction format

- If D = 1, then register operand existing in byte-2 is the destination operand, otherwise (i.e., if D = 0) it is a source operand.
- W indicates whether the operation is an 8-bit or a 16-bit data. If W = 0 then it is an 8-bit operation, else (i.e., W = 1) it is 16-bit one.
- The 2nd byte (byte-2) indicates whether one of the operands is in memory or both are in registers. This byte contains three fields:

Field	Abbreviation	Length (no. of bits)
Mode field	MOD	2
Register field	REG	3
Register/Memory field	r/m	3

33. Discuss the MOD and r/m and REG fields.

Ans. MOD field is a 2-bit field. It addresses memory in the following manner.

MOD field values

0 0 ⟶ Memory addressing without displacement
0 1 ⟶ Memory addressing with 8-bit displacement
1 0 ⟶ Memory addressing with 16-bit displacement
1 1 ⟶ Register addressing with
W = 0 for 8-bit data
and W = 1 for 16-bit data

The r/m field which is a 3-bit field, along with MOD field defines the 2nd operand. If MOD = 11, then it is a register to register mode. Again if MOD = 00,01 or 10 then it is a memory mode. Table 11.5 shows how effective address of memory operand gets selected for MOD = 00,01 and 10 values.

Table 11.5: For effective address calculation, values of MOD and r/m

r/m	MOD 00	MOD 01	MOD 10	MOD 11	
				W = 0	W = 1
000	[BX] + [SI]	[BX]+[SI]+D8	[BX]+[SI]+D16	AL	AX
001	[BX] + [DI]	[BX]+[DI]+D8	[BX]+[DI]+D16	CL	CX
010	[BP] + [SI]	[BP]+[SI]+D8	[BP]+[SI]+D16	DL	DX
011	[BP] + [DI]	[BP]+[DI]+D8	[BP]+[DI]+D16	BL	BX
100	[SI]	[SI]+D8	[SI]+D16	AH	SP
101	[DI]	[DI]+D8	[DI]+D16	CH	BP
110	Direct Addressing	[BP]+D8	[BP]+D16	DH	SI
111	[BX]	[BX]+D8	[BX]+D16	BH	DI

Again, for MOD values 00,01 and 10, the default segment registers selected are shown in Table 11.6.

Table 11.6: Segment register for various memory addressing modes

r/m	MOD 00	MOD 01	MOD 10	Segment Register used
000	[BX] + [SI]	[BX]+[SI]+DS	[BX]+[SI]+D16	DS
001	[BX] + [DI]	[BX]+[DI]+DS	[BX]+[DI]+D16	DS
010	[BP] + [SI]	[BP]+[SI]+DS	[BP]+[SI]+D16	SS
011	[BP] + [DI]	[BP]+[DI]+DS	[BP]+[DI]+D16	SS
100	[SI]	[SI]+DS	[SI]+D16	DS
101	[DI]	[DI]+DS	[DI]+D16	DS
110	D16 Direct Addressing [DS]	[BP]+DS Stack pointer register [SS]	[BP]+D16 Stack segment register [SS]	DS or SS as in MOD column
111	[BX]	[BX]+DS	[BX]+D16	DS

The REG field is a 3-bit field and indicates the register for the first operand which can be source/destination operand, depending on D = 0/1.

How REG field along with the status of W(0 or 1) select the different registers, is shown in Table 11.7.

Table 11.7: Definition of registers with 'W'

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

34. Discuss the instruction format for segment override prefix.

Ans. Default segment selection can be overriden by the override prefix byte, as shown in below.

0	0	1	r	r	1	1	0

Selects Segment

Depending on the 2-bit rr values, the segments selected are shown in Table 11.8.

Table 11.8 : Segment selection by override prefix technique

rr values	Segments selected
00	ES
01	CS
10	SS
11	DS

The override prefix byte follows the opcode byte of the instruction, whenever used.

35. Is direct memory to memory data transfer possible in 8086?

Ans. No, 8086 does not have provision for direct memory to memory data transfer.

For this to be implemented, AX is used as an intermediate stage of data. The source byte (from the memory) is moved into AX register with one instruction. The second instruction moves the content of AX into destination location (into another memory location). As example,

MOV AH, [SI]
MOV [DI], AH

Here, the first instruction moves the content of memory location, whose offset address remains in SI, into AH. The second instruction ensures that the content of AH is moved into another memory location whose offset address is in DI.

36. Can the data segment (DS) register be loaded directly by its address?

Ans. No, it cannot be done directly. Instead, AX is loaded with the initial address of the DS register and then it is transferred to DS register, as shown below:

MOV AX, DS ADDR:	AX is loaded with initial address of DS register
MOV DS, AX:	DS is loaded with AX, i.e., ultimately with DS ADDR

37. Show, in tabular form, the default and alternate segment registers for different types of memory references.

Ans. Table 11.9 shows the default and alternate register segments which can be used for different types of memory references.

Table 11.9: Default and alternate register assignments

Type of memory reference	Default segment	Alternative segment	Offset (Logical address)
Instruction fetch	CS	None	IP
Stack operation	SS	None	SP, BP
General data	DS	CS, ES, SS	Effective address
String source	DS	CS, ES, SS	SI
String destination	ES	None	DI
BX used as pointer	DS	CS, ES, SS	Effective Address
BP used as pointer	SS	CS, ES, DS	Effective Address

12

Memory Organisation

1. Mention the address capability of 8086 and also show its memory map.

Ans. 8086, via its 20-bit address bus, can address 2^{20} = 1,048,576 or 1 MB of different memory locations. Thus the memory space of 8086 can be thought of as consisting of 1,048,576 bytes or 524,288 words.

The memory map of 8086 is shown in Fig. 12.1, where the whole memory space starting from 00000 H to FFFFF H is divided into 16 blocks—each one consisting of 64 KB. This division is arbitrary but at the same time a convenient one—because the most significant hex digit increases by 1 with each additional block. Thus, 30000 H memory location is 65,536 bytes higher in memory than the memory location 20000 H.

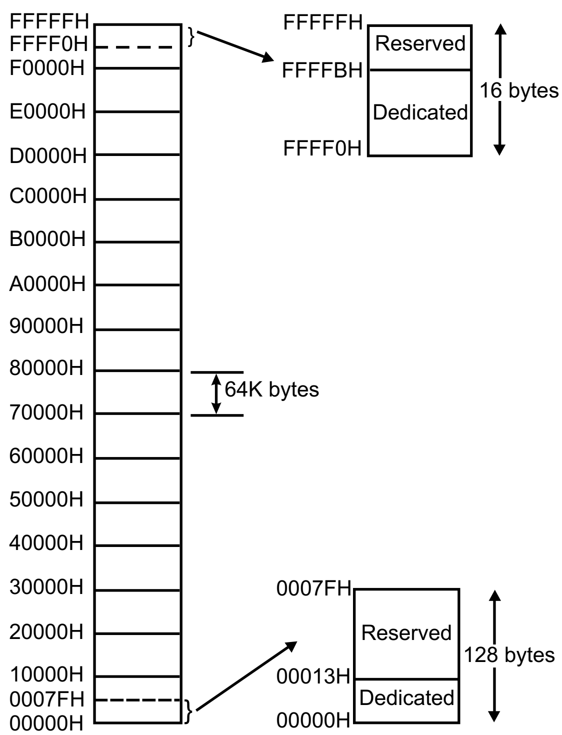


Fig. 12.1: Memory map for the 8086 microprocessor. Some memory locations are dedicated or reserved.

The lower and upper ends of the memory map are shown separately—earmarking some spaces as reserved and some as 'dedicated'.

The reserved locations are meant for future hardware and software needs while the dedicated locations are used for processing of specific system interrupts and reset functions.

2. Mention the different types of memory segmentations of 8086.

Ans. The different memory segmentations done in case of 8086 are

- Continuous
- partially overlapped
- fully overlapped and
- disjointed

This is shown in Fig.12.2.

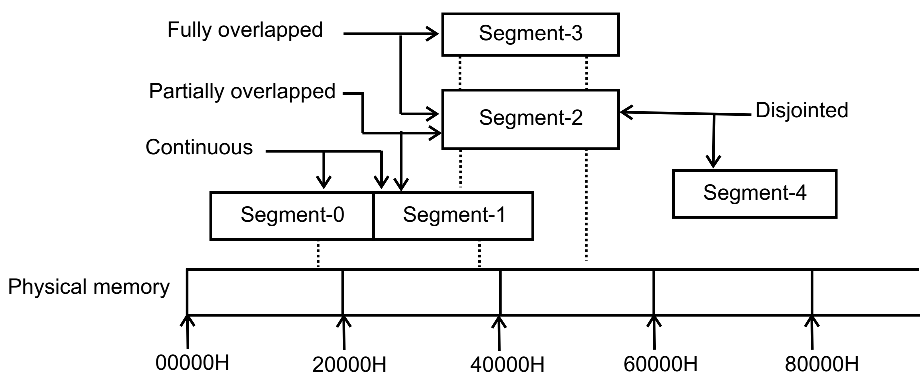


Fig. 12.2: Depiction of different types of segments

In the figure,

Segments-0	and	1 ⟶	Continuous
Segments-1	and	2 ⟶	Partially overlapped
Segments-2	and	3 ⟶	Fully overlapped
and Segments-2	and	4 ⟶	Disjointed

3. Describe memory segmentation scheme of 8086. What is meant by currently active segments?

Ans. 1 MB memory of 8086 is partitioned into 16 segments—each segment is of 64 KB length. Out of these 16 segments, only 4 segments can be active at any given instant of time—these are code segment, stack segment, data segment and extra segment. The four memory segments that the CPU works with at any time are called currently active segments. Corresponding to these four segments, the registers used are Code Segment Register (CS), Data Segment Register (DS), Stack Segment Register (SS) and Extra Segment Register (ES) respectively.

Each of these four registers is 16-bits wide and user accessible—i.e., their contents can be changed by software.

The code segment contains the instruction codes of a program, while data, variables and constants are held in data segment. The stack segment is used to store interrupt and subroutine return addresses.

The extra segment contains the destination of data for certain string instructions. Thus 64 KB are available for program storage (in CS) as well as for stack (in SS) while 128 KB of space can be utilised for data storage (in DS and ES).

One restriction on the base address (starting address) of a segment is that it must reside on a 16-byte address memory—examples being 00000 H, 00010 H or 00020 H, etc.

4. Mention the maximum size of memory that can be active for 8086.

Ans. The maximum size of active memory for 8086 is 256 KB. The break-up being
64 KB for program
64 KB for stack and
128 MB for data.

5. Why memory segmentation is done for 8086?

Ans. Memory segmentation, as implemented for 8086, gives rise to the following advantages:

- Although the address bus is 20-bits in width, memory segmentation allows one to work with registers having width 16-bits only.
- It allows instruction code, data, stack and portion of program to be more than 64 KB long by using more than one code, data, extra segment and stack segment.
- In a time-shared multitasking environment when the program moves over from one user's program to another, the CPU will simply have to reload the four segment registers with the segment starting addresses assigned to the current user's program.
- User's program (code) and data can be stored separately.
- Because the logical address range is from 0000 H to FFFF H, the same can be loaded at any place in the memory.

6. Discuss logical address, base segment address and physical address.

Ans. The logical address, also goes by the name of effective address or offset address (also known as offset), is contained in the 16-bit IP, BP, SP, BX, SI or DI.

The 16-bit content of one of the four segment registers (CS, DS, ES, SS) is known as the base segment address.

Offset and base segment addresses are combined to form a 20-bit physical address (also called real address) that is used to access the memory. This 20-bit physical address is put on the address bus ($AD_{19} – AD_0$) by the BIU.

7. Describe how the 20-bit physical address is generated.

Ans. The 20-bit physical (real) address is generated by combining the offset (residing in IP, BP, SP, BX, SI or DI) and the content of one of the segment registers CS, DS, ES or SS. The process of combination is as follows:

The content of the segment register is internally appended with 0 H (0000 H) on its right most end to form a 20-bit memory address—this 20-bit address points to the start of the segment. The offset is then added to the above to get the physical address.

Fig. 12.3 shows pictorially the actual process of generating a 20-bit physical address.

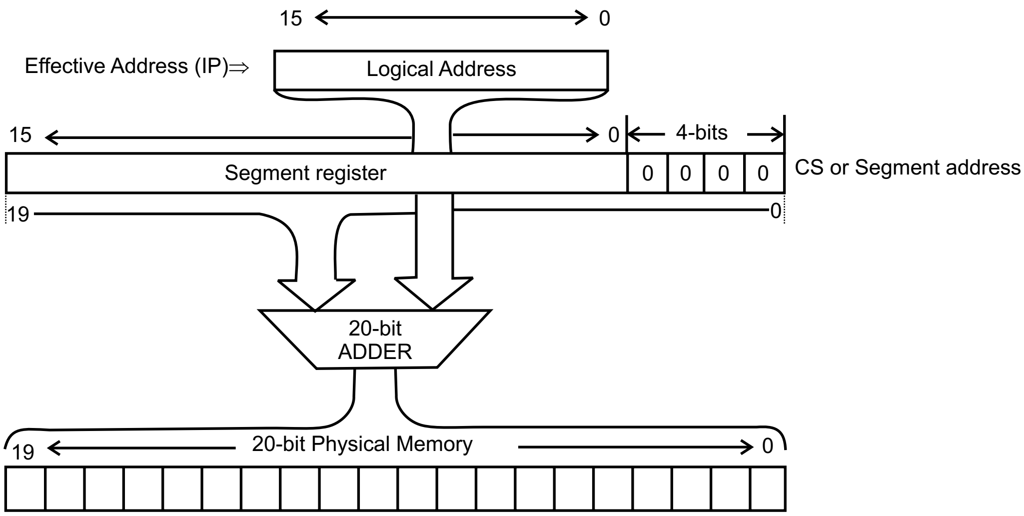


Fig. 12.3: Physical address generation

Thus, Physical Address = Segment Register content 16 D + Offset.

8. Although 8086 is a 16-bit μP, it deals with 8-bit memory. Why?

Ans. This is so for the following two reasons:

- It enables the microprocessor to work with both on bytes and words. This is very important because many I/O devices such as printers, terminals, modems etc, transfer ASCII coded data (7 or 8 bits).
- Quite a few of the operation codes of 8086 are single bytes while so many other instructions are there which vary from 2 to 7 bytes. By working with byte-width memory, these varied opcodes can easily be handled.

9. Is the flat scheme of memory applied for 8086 μP?

Ans. No, the flat (or unsegmented) scheme of memory is not applied for 8086 μP, because the memory of the same is a segmented one. In flat scheme, the entire memory space is thought of as a single addressable memory unit.

The flat scheme can be applied for 8086 by initialising all the segment registers with identical (or same) base address. Then all memory operations will refer to the same memory space.

10. Describe how memory is organised for 8086 μP?

Ans. The total address space 1 MB of 8086 is divided into 2 banks of memory—each bank of maximum size 512 KB. One is called the high order memory bank (or high bank) and the other low order memory bank (or low bank).

Low bank, high bank or both banks can be accessed by utilizing two signals $\overline{BHE}$ and A_0. Table 12.1 shows the three possible references to memory.

Table 12.1: Memory references

$\overline{BHE}$	A_0	**Processing**
0	0	Both Banks Active 16-bit world transfer on $AD_{15} \Leftrightarrow AD_0$
0	1	Only High bank Active (One byte from/to odd address on $AD_{15} \Leftrightarrow AD_8$)
1	0	Only Low bank Active (One byte from/to even address on $AD_7 \Leftrightarrow AD_0$)
1	1	No Bank Active

The high bank is selected for A_0=1 and $\overline{BHE}$ =0 and is connected to D_{15}–D_8 while the low bank is selected for A_0=0 and $\overline{BHE}$ =1. Neither low bank nor high bank would selected for A_0=1 and $\overline{BHE}$ =1.

Fig. 12.4 shows how the total address space (1MB) of 8086 is physically implemented by segregating it into low and high banks. It also shows that $\overline{CS}$ signal of the high bank is connected to $\overline{BHE}$ while the $\overline{CS}$ signal of the low bank is connected to A_0.

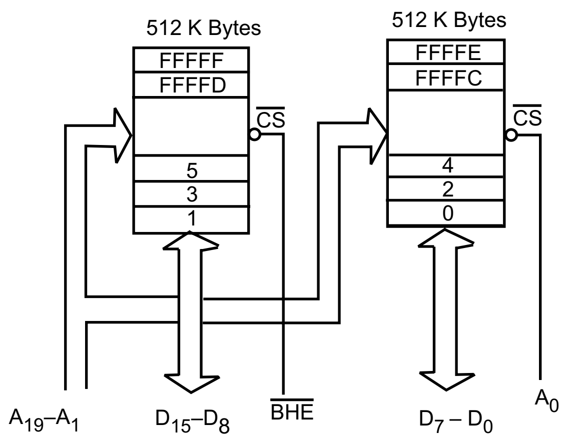


Fig.12.4: Selection of high and low banks of 8086

11. Show the profiles of low and high order memory banks.

Ans. The low and high order memory banks correspond to even and odd banks respectively.

The $\overline{CS}$ signal of low order memory bank is selected when $\overline{CS}$ = 0. Since A_0 (lowest address bus line) is connected to $\overline{CS}$, hence A_0 must have be to low for the low order bank to be selected. That is why the low order bank corresponds to even bank. Similarly the high order bank is selected when A_0 = 1. Hence, the higher order bank is called odd bank.

The profile of the low and high order banks are shown below in Fig. 12.5

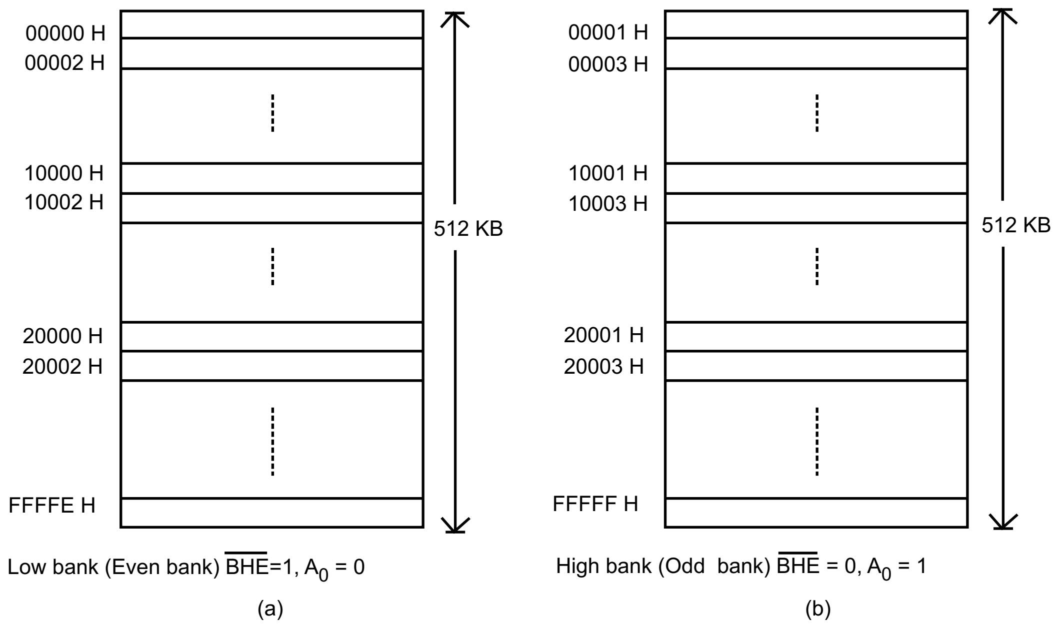


Fig. 12.5: (a) Low or even bank (b) High or odd bank

12. Draw the diagrams of (a) even-addressed byte transfer (b) odd-addressed byte transfer (c) even-addressed word transfer and (d) odd-addressed word transfer.

Ans. Fig. 12.6 shows the above four cases. A, B are representing the addresses while (A), (B) represent the content of address locations A and B respectively.

Figures (a) and (b) correspond to byte transfers for even and odd-addressed memory locations respectively. The shaded memory location indicates that the content of that particular memory location comes out either via higher byte data bus (D_{15}–D_8) or lower byte data bus (D_7–D_0) respectively.

Figures (a), (b) and (c) complete the data transfer in one bus cycle only.

$$\text{For Figure (a), } \overline{BHE} = 1, A_0 = 0$$

$$\text{For Figure (b), } \overline{BHE} = 0, A_0 = 1$$

$$\text{For Figure (c), } \overline{BHE} = 0, A_0 = 0$$

Figure (d) corresponds to an odd-addressed word transfer and it takes two bus cycles to complete this transfer.

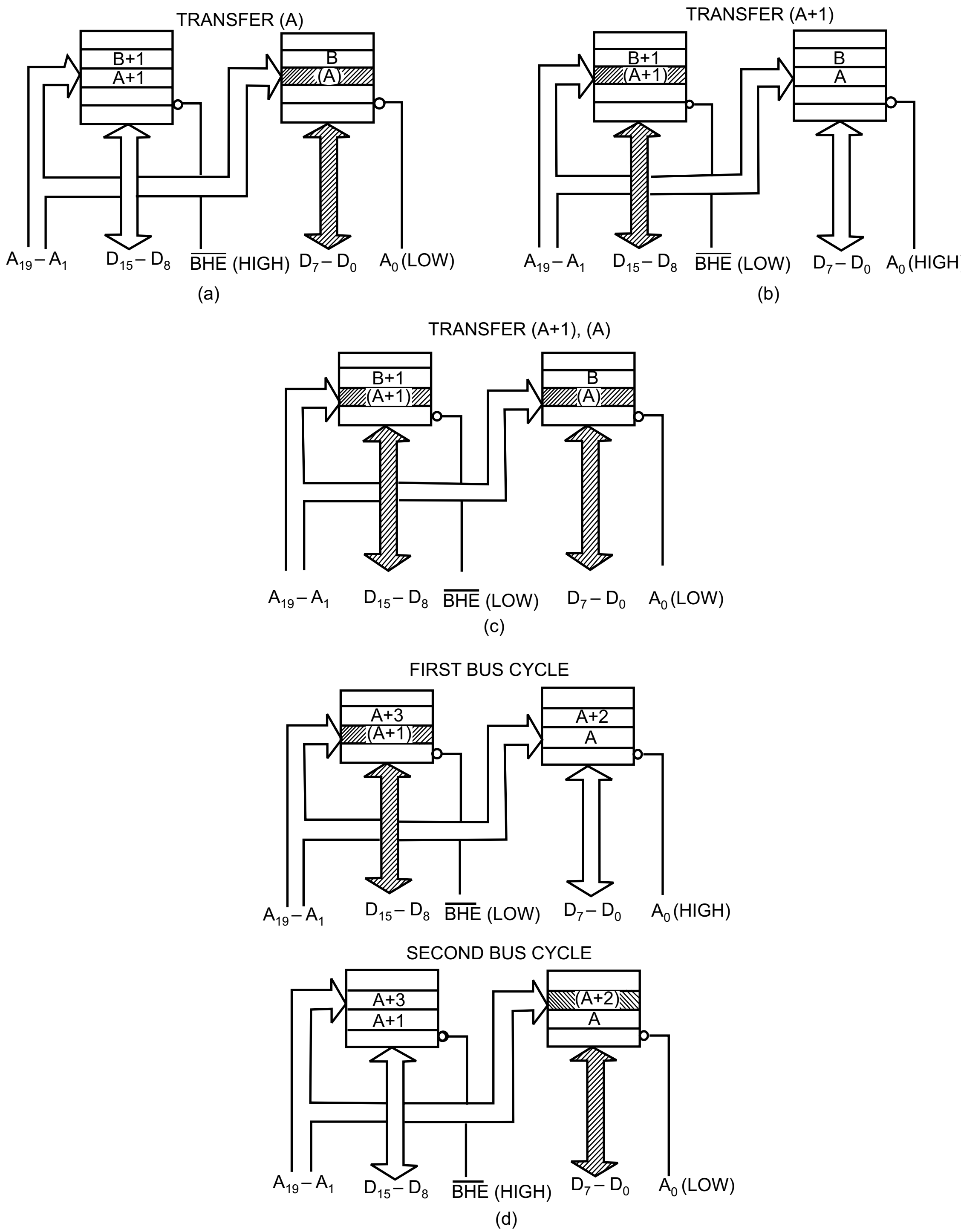


Fig. 12.6: (a) Even-addressed byte transfer by the 8086. (Reprinted with permission of Intel Corporation, © 1979)
(b) Odd-addressed byte transfer by the 8086. (Reprinted with permission of Intel Corporation, © 1979)
(c) Even-addressed word transfer by the 8086. (Reprinted with permission of Intel Corporation, © 1979)
(d) Odd-addressed word transfer by the 8086. (Reprinted with permission of Intel Corporation, © 1979)

This odd-addressed word is an unaligned one and the LSB of the address is in the high memory bank.

The odd byte of the word is at address location A + 1 and is selected by making $\overline{BHE}$ = 0 and A_0. Thus in the first bus cycle, data to transferred on $D_{15} - D_8$.

In the second bus cycle, 8086 automatically increments the address. Hence A_0 becomes 0, representing even address A + 2. This is in the low bank and is accessed by making $\overline{BHE}$ = 1 and $A_0 = 0$.

13. Which pins identify the segment registers used for 20-bit physical address generation?

Ans. Pins A_{16} and A_{17} become S_3 and S_4 from the second bus cycle. This 2-bit combination of S_3 and S_4 indicate the segment register used for physical address generation and is shown in Table 12.2

Table 12.2: Identifying the segment register used for 20-bit physical address generation

S_4	S_3	Segment Register
0	0	Extra
0	1	Stack
1	0	Code/none
1	1	Data

The two status codes are output both in the maximum and minimum mode.

14. What is the maximum size of the memory that can be accessed by 8086?

Ans. The two status codes S_4 and S_3 together point to the segment register used for 20-bit physical address generation and can be examined by external circuitry to enable separate 1 MB address space for each of CS, ES, DS, and SS. This would enable memory address to be expanded to a maximum of 4MB for 8086 µP.

15. Draw the Read and Write bus cycles for 8086 µP in Minimum mode.

Ans. Fig. 12.7 shows the Read and Write bus cycles for 8086 µP in the Minimum mode.

The bus cycle consists of 4T states. ALE signal stays high for T_1 state at the end of which it goes low which is utilised by latches to latch the address. Hence, during $T_2 - T_4$ states, $AD_{15} - AD_0$ lines act as data lines. The $M/\overline{IO}$, $\overline{RD}$ and $\overline{WR}$ signals can be combined to generate individual $\overline{IOR}$, $\overline{IOW}$ and $\overline{MEMR}$, $\overline{MEMW}$ signals.

The Read and Write cycles show that data are made available during T_3 and T_2 states respectively.

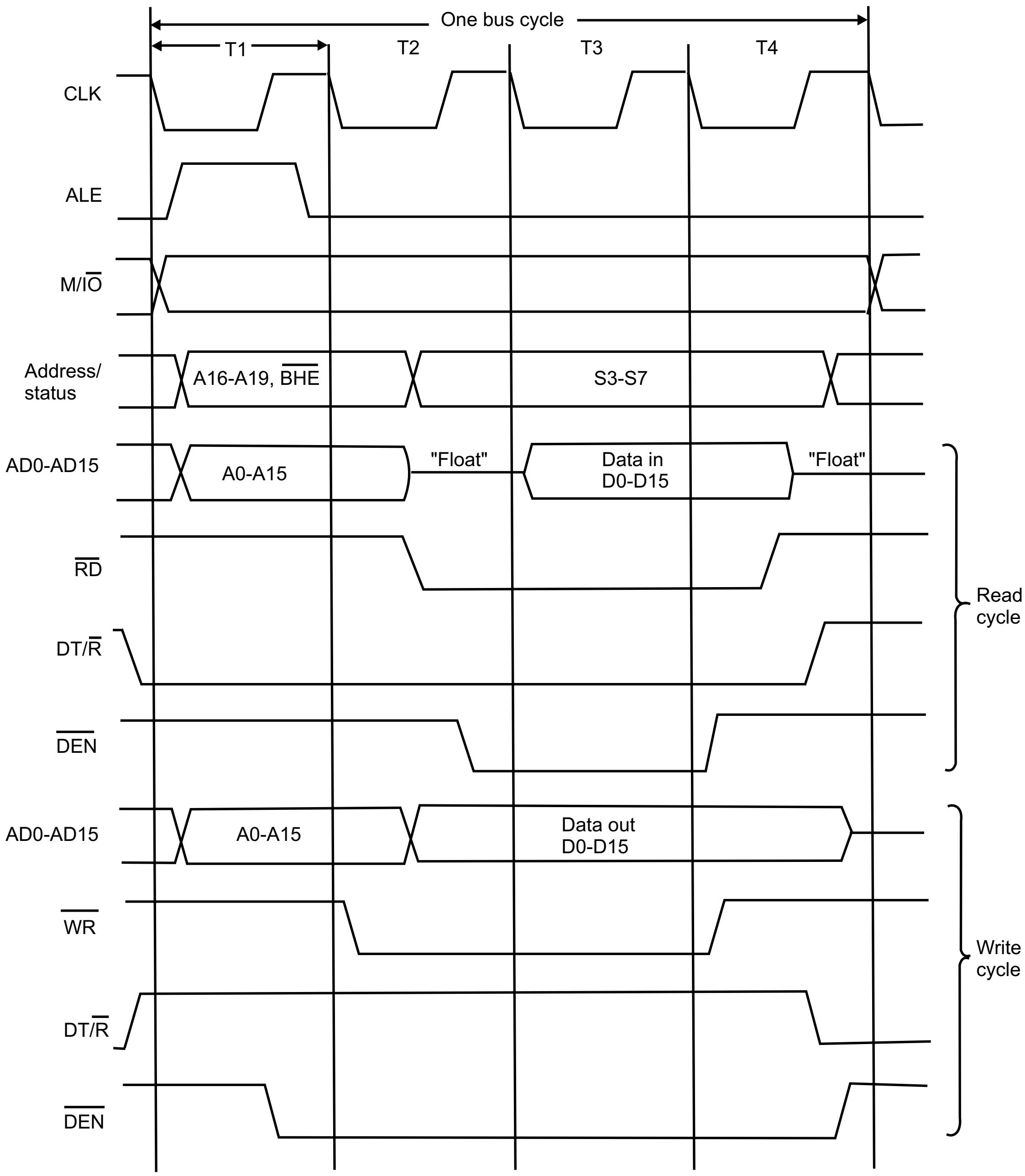


Fig.12.7: 8086 microprocessor read and write bus cycles.The address lines are valid during the T1 state but become the data lines and status indicators during T2–T4

13

Addressing Modes of 8086

1. What is meant by addressing mode?

Ans. An instruction consists of an opcode and an operand. The operand may reside in the accumulator, or in a general purpose register or in a memory location.

The manner in which an operand is specified (or referred to) in an instruction is called addressing mode.

2. Name the different addressing modes of 8086.

Ans. The following are the different addressing modes of 8086:

- Register operand addressing.
- Immediate operand addressing.
- Memory operand addressing.

3. Mention the different memory addressing modes.

Ans. The different memory addressing modes are:

- Direct Addressing
- Register Indirect Addressing
- Based Addressing
- Indexed Addressing
- Based Indexed Addressing and
- Based Indexed with displacement.

4. How the physical address is generated for the different memory addressing modes?

Ans. Physical address (for the operand) is the address from which either a read or write operation is initiated. The following shows the manner of generation of physical address.

$$\begin{aligned}\text{Physical address} &= \text{Segment base : Effective address}\\ &= \text{Segment base : Base + Index + Displacement}\end{aligned}$$

$$= \begin{Bmatrix} \text{CS} \\ \text{SS} \\ \text{DS} \\ \text{ES} \end{Bmatrix} : \begin{Bmatrix} \text{BX} \\ \text{BP} \end{Bmatrix} + \begin{Bmatrix} \text{SI} \\ \text{DI} \end{Bmatrix} + \begin{Bmatrix} \text{8-bit Displacement} \\ \text{Or} \\ \text{16-bit Displacement} \end{Bmatrix}$$

i.e., effective address, for its generation, can have as many as three elements: Base, Index and Displacement. Thus the effective address is generated from the following:

Effective address = Base + Index + Displacement

The segment registers can be CS, SS, DS or ES. Base can be BX or PB. Index can be SI or DI and the Displacement can either be 8-bit or 16-bit.

It should be noted that not all the three elements viz., base, index or displacement are always used for effective address calculations.

5. Give examples each of (a) Register Addressing mode (b) Immediate Addressing mode.

Ans. (a) Register Addressing Mode.

In this mode, either an 8-bit or a 16-bit general purpose register contains the operand. Some examples are:

```
MOV AX, BX
MOV CX, DX
ADD AL, DH
ADD DX, CX
```

The content of BX register is moved to AX register in the first example, while in the third example, content of DH is added to the content of AL.

Here, source and destination of data are CPU registers.

(b) Immediate Addressing Mode.

In this mode, the operand is contained in the instruction itself, i.e., the operand forms a part of the instruction itself. The operand can be either 8-bit or 16-bit in length. Some examples are:

```
MOV AL, 83 H
ADD AX, 1284 H
```

In the first example, 83 H is moved to AL register while in the second example, 1284 H is added to the contents of AX register. Here, source of data is within the instruction.

6. Discuss the Direct Addressing Mode.

Ans. In a way it is similar to 'Immediate Addressing Mode'. In Immediate Addressing Mode, data follows the instruction opcode, while in this case an effective address follows the same. Thus in this case:

PA = Segment base : Direct address.

By default, the segment base register is DS. Thus,

$$PA = DS : EA$$

But if a segment override prefix (SEG) is used in the instruction, then any one of the four segment registers can be referenced. Hence, in general,

$$PA = \begin{Bmatrix} CS \\ DS \\ SS \\ ES \end{Bmatrix} : \text{Direct Address}$$

As an example, MOV CX, [ALPHA]

It means, "move the contents of the memory location, which is labelled as ALPHA in the current data segment, into register CX".

Thus, if DS = 0300 H, and value assigned to ALPHA is 3216 H, then

$$\begin{aligned} PA &= 03000\text{ H} + 3216\text{ H} \\ &= 06216\text{ H} \end{aligned}$$

Thus, data contained in address locations 06217 H and 06216 H will be stored in CH and CL registers respectively.

MOV [0404 H], CX would move the contents of CL to offset address 0404 H (relative to data segment register DS) and CH to 0405 H. Here, memory address is supplied within the instruction.

7. Discuss Register Indirect Addressing Mode.

Ans. In a way, this mode of addressing is similar to direct addressing mode in the sense that content of DS is combined with the effective address to get the physical address.

But the difference lies in the manner in which the offset is specified. In direct addressing mode EA is constant while in this mode EA is a variable.

EA can reside in either base register (BX or BP) or index register (SI or DI). The default segment register is DS, but again by using a segment override prefix (SEG), any of the four segment registers can be referenced.

Thus, PA can be computed as:

$$PA = \begin{Bmatrix} CS \\ DS \\ SS \\ ES \end{Bmatrix} : \begin{Bmatrix} BX \\ BP \\ SI \\ DI \end{Bmatrix}$$

An instruction of this mode of addressing is:

MOV CX, [SI]

Execution of this instruction entails moving the content of the memory location having its offset value in SI from the beginning of the current data segment to the CX register.

If DS = 0300 H and SI = 3216 H, then PA becomes

PA = 03000 H + 3216 H = 06216 H.

Thus data contained in 06217 H and 06216 H will be placed in CH and CL registers respectively. Here, memory address is supplied in an index or pointer register.

8. Discuss Based Addressing Mode.

Ans. The physical address in this case is generated as follows:

PA = Segment base: Base + Displacement

$$PA = \begin{Bmatrix} CS \\ DS \\ SS \\ ES \end{Bmatrix} : \begin{Bmatrix} BX \\ BP \end{Bmatrix} + \begin{Bmatrix} \text{8-bit Displacement} \\ \text{or} \\ \text{16-bit Displacement} \end{Bmatrix}$$

i.e., the physical address is generated by adding either an 8-bit or 16-bit displacement to the contents of either base register BX or base pointer register BP and the current value in DS or SS respectively.

Fig. 13.1 shows the utility of using either BX/BP and displacement. The figure shows a data structure starting from 'Element 0' to 'Element n'. Inserting a zero value for displacement would ensure accessing 'Element 0' of the structure. For accessing different elements within the same data structure, all that is to be done is to change the value of displacement.

Whereas, to access the same element is another data structure the value of the base register has to be changed, keeping the value of displacement same as before.

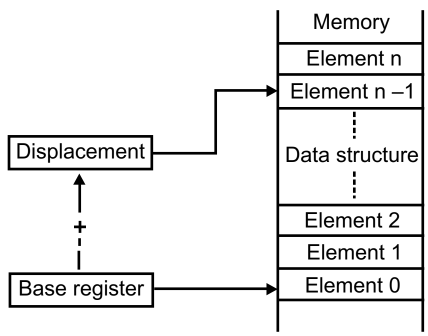


Fig.13.1: Based addressing mode of a structure of data

An example of this mode of addressing is MOV [BX] + ALPHA, AH

Here, ALPHA denotes displacement (which can be 8 or 16-bits) and BX the base register. Together, they give EA of the destination operand.

If DS = 3000 H, BX = 1234 H and displacement (16-bit) = 0012 H,

Then, PA = 03000 H + 1234 H + 0012 H = 04246 H

Thus the content of AH (source operand) is placed in the physical address 04246 H (destination operand memory location).

In this mode also, the default register i.e., DS can be changed by a segment override prefix (SEG). Also for accessing data from the stack segment of the memory, BP is to be used instead of BX.

Here, the memory address is the sum of BX or BP base registers plus an 8 or 16-bit displacement specified in the instruction.

9. Discuss Indexed Addressing Mode.

Ans. In a way, this mode of addressing is similar to the based addressing mode but the jobs carried out by base register and displacement in the based addressing mode are done by displacement and index register respectively in indexed addressing mode and shown in Fig. 13.2.

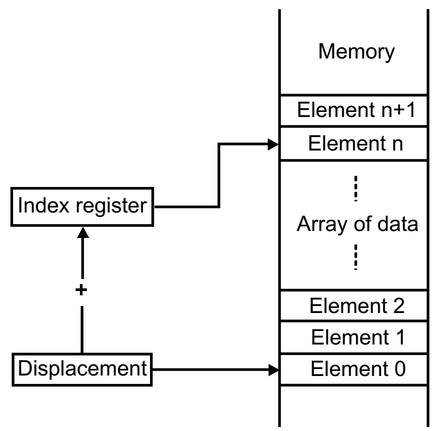


Fig.13.2: Indexed addressing mode of an array of data elements

The physical address in this case is generated as follows:
PA = Segment Base : Index + Displacement

$$PA = \begin{Bmatrix} CS \\ DS \\ SS \\ ES \end{Bmatrix} : + \begin{Bmatrix} SI \\ DI \end{Bmatrix} + \begin{Bmatrix} \text{8-bit Displacement} \\ \text{Or} \\ \text{16-bit Displacement} \end{Bmatrix}$$

An example of this mode of addressing is:
MOV [SI] + ALPHA, AH
where, ALPHA represents displacement.
Assuming, DS = 3000 H, SI = 1234 H and ALPHA (displacement) = 0012 H.
Thus, PA = 03000 H + 1234 H + 0012 H = 04246 H.

Thus, the data value residing in source operand AH will be moved to the physical address location 04246 H. Here, memory address is the sum of the index register plus an 8 or 16-bit displacement specified in the instruction.

10. Discuss Based Indexed Addressing mode.

Ans. It is a combination of based and indexed addressing modes. The physical address is generated in this case in the following manner:

PA = Segment Base : Base + Index

$$= \begin{Bmatrix} CS \\ DS \\ ES \\ SS \end{Bmatrix} : \begin{Bmatrix} BX \\ BP \end{Bmatrix} + \begin{Bmatrix} SI \\ DI \end{Bmatrix}$$

Here, BX and BP registers are used for data and stack segments respectively.
An example of this mode of addressing is as follows:

MOV AL, [BX] [SI]

If DS = 3000 H, BX = 1000 H and SI = 1234 H, then

PA = 03000 H + 1000 H + 1234 H = 05234 H

On executing this instruction, the value stored in memory location 05234 H will be stored in AL.

Here, memory address is the sum of an index register and a base register.

11. Discuss Based Indexed with displacement Addressing Mode.

Ans. It is a combination of based addressing mode and indexed addressing mode along with an 8 or 16-bit displacement. The physical address is generated in the following manner:

PA = Segment base : Base + Index + Displacement

$$PA = \begin{Bmatrix} CS \\ DS \\ ES \\ SS \end{Bmatrix} : \begin{Bmatrix} BX \\ BP \end{Bmatrix} + \begin{Bmatrix} SI \\ DI \end{Bmatrix} + \begin{Bmatrix} \text{8-bit displacement} \\ \text{or} \\ \text{16-bit displacement} \end{Bmatrix}$$

This addressing mode is used to access a two dimensional (m × n) array, as shown in Fig. 13.3. The displacement, having a fixed value, locates the starting position of the array in the memory while the base register specifies one coordinate (say m) and index register the other coordinate (say n). Any position in the array can be located simply by changing the values in the base and index registers.

An example of this mode of addressing is as follows:

MOV AL, [BX] [SI] + ALPHA

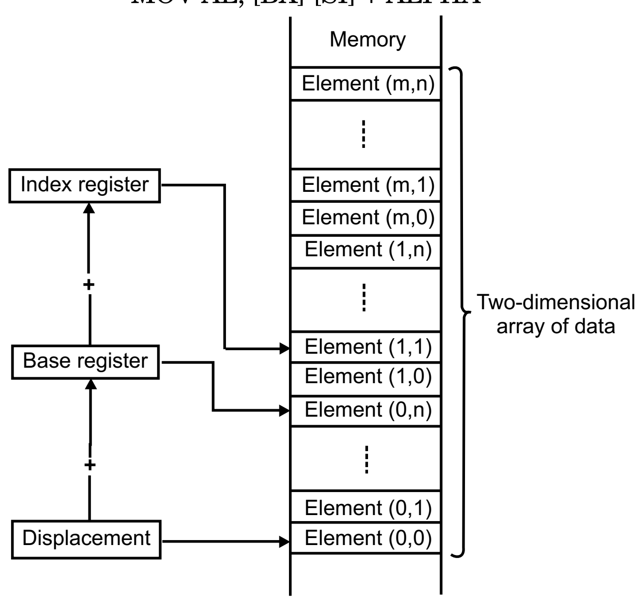


Fig.13.3: Based indexed with displacement addressing mode of a two-dimensional array of data

Thus the offset or effective address can be calculated from the contents of base and index registers and the fixed displacement as represented by ALPHA.

Assuming: DS = 3000 H, BX = 1000 H, SI = 1234 H and ALPHA (displacement) = 0012 H.

Thus, PA = 03000 H + 1000 H + 1234 H + 0012 H = 05246 H.

On executing this instruction, the value stored in memory location 05246 H (source operand) will be stored in AL. Here, memory address is the sum of an index register, a base register and an 8 or 16-bit displacement within the instruction.

14

The Instruction Set of 8086

1. How many instructions are there in the instruction set of 8086?

Ans. There are 117 basic instructions in the instruction set of 8086.

2. Do 8086 and 8088 have the same instruction set?

Ans. Yes, both 8086 and 8088 have the same instruction set.

3. Mention the groups in which the instruction set of 8086 can be categorised.

Ans. The instruction set of 8086 can be divided into the following number of groups, namely:

- Data transfer instructions
- Arithmetic instructions
- Logic instructions
- Shift instructions
- Rotate instructions
- Flag control instructions
- Compare instructions
- Jump instructions
- Subroutines and subroutine handling instructions
- Loop and loop handling instructions
- Strings and string handling instructions.

4. Mention the different types of data transfer instructions.

Ans. The different types include:

- Move byte or word instructions (MOV)
- Exchange byte or word instruction (XCHG)
- Translate byte instructions (XLAT)
- Load effective address instruction (LDA)
- Load data segment instruction (LDS)
- Load extra segment instruction (LES)

5. Can the MOV instruction transfer data directly between a source and destination that both reside in external memory?

Ans. No, it cannot. With the first MOV instruction, data from the source memory is to be moved into an internal register-normally accumulator. The second MOV instruction places the accumulator content into the destination memory.

6. Explain the following two examples:

(a) MOV CX, CS

(b) MOV AX, [ALPHA]

Ans. **(a) MOV CX, CS:** It stands for, "move the contents of CS into CX". If CS contains 1234 H, then on execution of this instruction, content of CX would become 1234 H i.e., content of CH = 12 H and Content CL = 34 H.

(b) MOV AX, [ALPHA]: Let, data segment register DS contains 0300 H and ALPHA corresponds to a displacement of 1234 H. Then the instruction stands for, "move the content of the memory location offset by 1234 H from the starting location 0300 H of the current data segment into accumulator AX". The physical address is

PA = 03000 H + 1234 H = 04234 H

Thus execution of the instruction results in content of memory location 04234 H is moved to AL and content of memory location 04235 H is moved to AH.

7. Show the forms of XCHG instruction and its allowed operands.

Ans. These are shown below in Fig. 14.1

Mnemonic	Meaning	Format	Operation	Flags affected
XCHG	Exchange	XCHG D,S	(D)↔(S)	None

(a)

Destination	Source
Accumulator Memory Register	Reg16 Register Register

(b)

Fig.14.1: (a) Exchange data transfer instruction (b) Allowed operands

8. Explain the instruction XCHG BX, CX.

Ans. The execution of this instruction interchanges the contents of BX and CX, i.e., original content of CX moves over to BX and original content of BX moves over to CX.

9. Explain XLAT instruction.

Ans. The translate (XLAT) data transfer instruction is shown in Fig.14.2. It can be used for say an ASCII to EBCDIC code conversion.

Mnemonic	Meaning	Format	Operation	Flags affected
XLAT	Translate	XLAT	((AL)+(BX)+(DS)O)→(AL)	None

Fig.14.2: Translate data transfer instruction

The content of BX represents the offset of the starting address of the look up table from the beginning of the current data segment while the content of AL represents the offset of the element which is to be accessed from the beginning of the look up table.

As an example, let DS = 0300 H, BX = 1234 H and AL = 05 H.

Hence, PA = 03000 H + 1234 H + 05 H = 04239 H

Thus, execution of XLAT would put the content of 04239 H into AL register.

Conceptually, the content of 04239 H in EBCDIC should be the same as the ASCII character equivalent of 05 H.

10. Explain the instruction LEA, LDS and LES.

Ans. These three instructions are explained in Fig.14.3. These instructions stand for load register with effective address (LEA), load register and data segment register (LDS) and load register and extra segment register (LES) respectively.

Mnemonic	Meaning	Format	Operation	Flags affected
LEA	Load effective address	LEA Reg 16, EA	(EA)→(Reg16)	None
LDS	Load register and DS	LDS Reg 16, Mem 32	(Mem 32)→(Reg16)	
			(Mem 32)→(Reg16)	
			(Mem 32+2)→(ES)	None
			(Mem 32+2)→(Reg 16)	
LES	Load register and ES	LES Reg 16, Mem 32	(Mem 32+2)→(ES)	None

Fig.14.3: LEA, LDS and LES data transfer instructions

LEA instruction loads a specified register with a 16-bit offset value. LDS and LES instructions load the specified register as well as DS or ES segment register respectively. Thus a new data segment will be activated by a single execution.

11. Indicate the different types of arithmetic instructions possible with 8086.

Ans. The different arithmetic instructions are addition, subtraction, multiplication and division and are shown in Fig.14.4.

	Addition
ADD	Add byte or word
ADC	Add byte or word with carry
INC	Increment byte or word by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
	Subtraction
SUB	Subtract byte or word
SBB	Subtract byte or word with borrow
DEC	Decrement byte or word by 1
NEG	Negate byte or word
AAS	ASCII adjust for subtraction
DAS	Decimal adjust for subtraction
	Multiplication
DIV	Divide byte or word unsigned
IDIV	Integer divide byte or word
AAD	ASCII adjust for division
CBW	Convert byte to word
CWD	Convert word to double word

Fig.14.4: Arithmetic instructions

12. Show the allowed operands for the instruction ADD, ADC and INC.

Ans. The allowed operands for ADD and ADC are shown in Fig.14.5 (a) and for INC it is shown in Fig.14.5 (b).

Destination	Source
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate
Accumulator	Immediate

(a)

Destination
Reg 16
Reg 8
Memory

(b)

Fig.14.5: (a) Allowed operands for ADD and ADC
(b) Allowed operands for INC

13. Show the different subtraction arithmetic instructions. Also show the allowed operands for (a) SUB and SBB (b) DEC and (c) NEG instructions.

Ans. The different subtraction arithmetic instructions and the allowed operands, for the different instructions are shown in Fig. 14.6 (a), (b), (c) and (d) respectively.

Mnemonic	Meaning	Format	Operation	Flags affected
SUB	Subtract	SUB D,S	(D)–(S)→(D) Borrow→(CF)	OF, SF, ZF, AF, PF, CF
SBB	Subtract with borrow	SBB D,S	(D)–(S)–(CF)→(D)	OF, SF, ZF, AF, PF, CF
DEC	Decrement by 1	DEC D	(D)–1→(D)	OF, SF, ZF, AF, PF
NEG	Negate	NEG D	0–(D)→(D) 1→(CF)	OF, SF, ZF, AF, PF, CF
DAS	Decimal adjust for subtraction	DAS		SF, ZF, AF, PF, CF OF undefined
AAS	AASCII adjust for subtraction	AAS		AF, CF, OF, SF, ZF, PF undefined

(a)

Destination	Source
Register	Register
Register	Memory
Memory	Register
Accumulator	Immediate
Register	Immediate
Memory	Immediate

(b)

Destination
Reg16
Reg 8
Memory

(c)

Destination
Register
Memory

(d)

Fig.14.6: (a) Subtraction arithmetic operations
(b) Allowed operands for SUB and SBB instructions
(c) Allowed operands for DEC instruction
(d) Allowed operands for NEG instruction

14. Show the different multiplication and division instructions and also the allowed operands.

Ans. The different multiplication and division instructions and also the allowed operands are shown in Fig.14.7 (a) and (b) respectively.

Mnemonic	Meaning	Format	Operation	Flags affected
MUL	Multiply (unsigned)	MUL S	(AL).(S8)→(AX) (AX)·(S16)→(DX).(AX)	OF, CF, SF, ZF, AF, PF, undefined
DIV	Division (unsigned)	DIV S	(1) Q((AX)/(S8))→(AL) R((AX)/(S8))→(AH) (2) Q((DX, AX)/(S16))→(AX) R((DX,AX)/(S16))→(DX) If Q is FF_{16} in case (1) or $FFFF_{16}$ in case (2), then type 0 interrupt occurs.	OF, SF, ZF, PF, CF undefined
IMUL	Integer multiply (signed)	IMUL S	(AL)·(S8)→(AX) (AX)·(S16)→(DX),(AX)	OF, CF, SF, ZF, AF, PF undefined
IDIV	Integer divide (signed)	IDIV S	(1) Q((AX)/(S8))→(AX) R((AX)/(S8))→(AH) (2) Q((DX,AX)/(S16))→(AX) R((DX,AX)(S16))→(DX) If Q is positive and exceeds $7FFF_{16}$, or if Q is negative and becomes less than 8001_{16}, then type 0 interrupt occurs	OF, SF, ZF, AF, PF, CF undefined
AAM	Adjust AL for multiplication	AAM	Q((AL)/10))→AH R((AL)/10))→AL	SF, ZF, PF, OF, AF, CF undefined
AAD	Adjust AX for division	AAD	(AH) · 10 + AL → AL 00 → AH	SF, ZF, PF, OF, AF, CF undefined
CBW	Convert byte to word	CWD	(MSB of AL)→(All bits of AH)	None
CWD	Convert word to double word	CWD	(MSB of AX)→(All bits of DX)	None

(a)

Source
Reg 8 Reg 16 Mem 8 Mem 16

(b)

Fig.14.7: (a) Multiplication and division instructions (b) Allowed operands.

15. Show the different logic instructions and also the allowed operands for (a) AND, OR and XOR (b) NOT instructions.

Ans. The different logic instructions as also the allowed operands for different instructions are shown in Fig.14.8 (a), (b) and (c) respectively.

Mnemonic	Meaning	Format	Operation	Flags affected
AND	Logical AND	AND D, S	(S) . (D) → (D)	OF, SF, ZF, PF, CF, AF undefined
OR	Logical Inclusive —OR	OR D, S	(S) + (D) → (D)	OF, SF, ZF, PF, CF, AF undefined
XOR	Logical Exclusive —OR	XOR D, S	(S) ⊕ (D) → (D)	OF, SF, ZF, PF, CF, AF undefined
NOT	Logical NOT	NOT D	($\overline{D}$) → (D)	None

(a)

Destination	Source
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate
Accumulator	Immediate

(b)

Destination
Register
Memory

(c)

Fig. 14.8: (a) Logic instructions
(b) Allowed operands for the AND, OR and XOR instructions
(c) Allowed operands for NOT instruction.

16. What are the two basic shift operations?

Ans. The two basic shift operations are logical shift and arithmetic shift.

The two logical shifts are shift logical left (SHL) and shift logical right (SHR), while the two arithmetic shifts are shift arithmetic left (SAL) and shift arithmetic right (SAR).

17. Show the different shift instructions and the allowed operands.

Ans. The various shift instructions and the allowed operands are shown in Fig. 14.9 (a) and (b) respectively.

Mnemonic	Meaning	Format	Operation	Flags affected
SAL/SHL	Shift arithmetic left/shift logic left	SAL/SHL D, Count	Shift the (D) left by the number of bit positions equal to Count and fill the vacated bit positions on the right with zeros.	CF, PF, SF, ZF, OF AF undefined OF undefined if count ≠ 1
SHR	Shift logical right	SHR D, Count	Shift the (D) right by the number of bit positions equal to Count and fill the vacated bits positions on the left with zeros.	CF, PF, SF, ZF, OF AF undefined OF undefined if count ≠ 1
SAR	Shift arithmetic right	SAR D, Count	Shift the (D) right by the number of bit positions equal to Count and fill the vacated bits positions on the left with original most significant bit.	OF, SF, ZF, CF, PF AF, undefined

(a)

Destination	Count
Register	1
Register	CL
Memory	1
Memory	CL

(b)

Fig.14.9: (a) Shift instructions, (b) Allowed operands

18. Show the different Rotate instructions and the allowed operands.

Ans. The different Rotate instructions and the allowed operands are shown in Fig. 14.10 (a) and (b) respectively.

Mnemonic	Meaning	Format	Operation	Flags affected
ROL	Rotate left	ROL D, Count	Rotate the (D) left by the number of bit positions equal to Count. Each bit shifted out from the left most bit goes back into the rightmost bit position.	CF OF underfined if count $\neq$ 1
ROR	Rotate right	ROR D, Count	Rotate the (D) right by the number of bit positions equal to Count. Each bit shifted out from the rightmost bit goes into the leftmost bit position.	CF OF undefined if count $\neq$ 1
RCL	Rotate left through carry	RCL D, Count	Same as ROL except carry is attached to (D) for rotation.	CF OF undefined if count $\neq$ 1
RCR	Rotate right through carry	RCR D, Count	Same as ROR except carry is attached to (D) for rotation.	CF OF undefined if count $\neq$ 1

(a)

Destination	Count
Register	1
Register	CL
Memory	1
Memory	CL

(b)

Fig. 14.10: (a) Rotate instructions (b) Allowed operands

19. Name the different flags control instructions, the operations performed by them and also the flags affected.

Ans. Fig. 14.11 shows the different flags control instructions, their meaning and the flags affected by respective instructions.

Mnemonic	Meaning	Operation	Flags affected
LAHF	Load AH from flags	(AH) ← (Flags)	None
SAHF	Store AH into flags	(Flags) ← (AH)	SF, ZF, AF, PF, CF
CLC	Clear carry flag	(CF) ← 0	CF
STC	Set carry flag	(CF) ← 1	CF
CMC	Complement carry flag	(CF) ← $(\overline{CF})$	CF
CLI	Clear interrupt flag	(IF) ← 0	IF
STI	Set interrupt flag	(IF) ← 1	IF

Fig. 14.11: Flag control instructions

20. Which register plays an important part in flag control instructions?

Ans. It is the accumulator register AH which plays an important part in flag control instructions.

For instance, if the present values in the flags are to be saved in some memory location then they are first to be loaded in the AH register and transfered to memory location, say M1, i.e.,

LAHF → Load AH from flags

MOV M1, AH → Move contents of AH into memory location M1.

As a second example, if the content of memory location, Say M2, is to be placed into flags, then

MOV AH, M2 → Move contents of memory location M2 into AH register.

SAHF → Store AH into flags.

21. List the characteristics of CMP instructions.

Ans. The following are the characteristics of CMP instruction:

- Can compare two 8-bit or two 16-bit numbers.
- Operands many reside in memory, a register in the CPU or be a part of an instruction.
- Results of comparison is reflected in the status of the six status flags—CF, AF, OF, PF, SF and ZF.
- CMP is a subtraction method—it uses 2's complement for this.
- Result of CMP is not saved—but based on CMP result, appropriate flags are either set/reset.

22. Explain CMP instruction.

Ans. The compare instruction, different operand combinations and the flags affected are shown in Fig. 14.12.

Mnemonic	Meaning	Format	Operation	Flags affected
CMP	Compare	CMP D, S	(D) – (S) is used in setting or resetting the flags	CF, AF, OF, PF, SF, ZF

(a)

Destination	Source
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate
Accumulator	Immediate

(b)

Fig.14.12: (a) Compare instruction (b) Operand combination

23. What are the two basic types of unconditional jumps? Explain.

Ans. The two basic types of unconditional jumps are intrasegment jump and intersegment jump.

The intrasegment jump is a jump for which the addresses must lie within the current code segment. It is achieved by only modifying the value of IP.

The intersegment jump is a jump from one code segment to another. For this jump to be effective, both CS and IP values are to be modified.

24. Show the unconditional jump instructions and the allowed operands.

Ans. The unconditional jump instruction, along with the allowed operands are shown in Fig. 14.13 (a) and (b) respectively.

Mnemonic	Meaning	Format	Operation	Affected Flags
JMP	Unconditional Jump	JMP Operand	Jump is initiated to the address specified by the operand	None

(a)

Operands
Short-label
Near-label
Far-label
Memptr 16
Regptr 16
Memptr 32

(b)

Fig.14.13: (a) Unconditional jump instruction (b) Allowed operands

Jump instructions carried out with a Short-label, Near-label, Memptr 16 or Regptr 16 type of operands represent intrasegment jumps, while Far-label and Memptr 32 represent intersegment jumps.

25. Distinguish between Short-label and Near-label jump instructions.

Ans. The distinction between the two is shown in a tabular form.

Short-label	Near-label
1. It specifies the jump relative to the address of the jump instruction itself. 2. Specifies a new value of IP with an 8-bit immediate operand. 3. Can be used only in the range of –126 to + 129 bytes from the location of the jump instruction.	1. It specifies the jump relative to the address of the jump instruction itself. 2. Specifies a new value of IP with a 16-bit immediate operand. 3. Can be used to cover the complete range of current code segment.

26. Describe the Memptr 16 and Regptr 16 jump instructions.

Ans. Both these types permit a jump to any location (address) in the current code segment. Again the contents of a memory or register indirectly specifies the address of the jump, as the case may be.

27. Show the conditional jump instruction and their different types.

Ans. The conditional jump instruction and their different types are shown in Fig. 14.14 (a) and (b) respectively.

Mnemonic	Meaning	Format	Operation	Flags affected
JCC	Conditional Jump	JCC Operand	If the specified condition CC is true the jump to the address specified by the operand is initiated; otherwise the next instruction is executed	None

(a)

Mnemonic	*Meaning*	*Condition*	*Mnemonic*	*Meaning*	*Condition*
JA	above	CF = 0 and ZF = 0	JAE	above or equal	CF = 0
JB	below	CF = 1	JBE	below or equal	CF = 1 or ZF = 1
JC	carry	CF = 1	JCXZ	CX register is zero	(CF or ZF) = 0
JE	equal	ZF = 1	JG	greater	ZF = 0 and SF = OF
JGE	greater or equal	SF = OF	JL	less	(SF xor OF) = 1
JLE	less or equal	((SF xor OF) or ZF) = 1	JNA	not above	CF = 1 or ZF = 1
JNAE	not above nor equal	CF = 1	JNB	not below	CF = 0
JNBE	not below nor equal	CF = 0 and ZF = 0	JNC	not carry	CF = 0
JNE	not equal	ZF = 0	JNG	not greater	((SF xor OF) or ZF) = 1
JNGE	not greater nor equal	(SF xor OF) = 1	JNL	not less	SF = OF
JNLE	not less nor equal	ZF = 0 and SF = OF	JNO	not overflow	OF = 0
JNP	not parity	PF = 0	JNS	not sign	SF = 0
JNZ	not zero	ZF = 0	JO	overflow	OF = 1
JP	parity	PF = 1	JPE	parity even	PF = 1
JPO	parity odd	PF = 0	JS	sign	SF = 1
JZ	zero	ZF = 1			

(b)

Fig. 14.14: (a) Conditional jump instruction (b) Types of conditional jump instructions

28. Show the subroutine CALL instruction and the allowed operands.

Ans. The CALL instruction and the allowed operands are shown in Fig.14.15(a) and (b) respectively.

Mnemonic	Meaning	Format	Operation	Flags affected
CALL	Subroutine call	CALL operand	Execution continues from the address of the subroutine specified by the operand. Information required to return back to the main program such as IP and CS are saved on the stack.	None

(a)

Operands
Near-proc Far-proc Memptr 16 Regptr 16 Memptr 32

(b)

Fig.14.15: (a) Subroutine call instruction (b) Allowed operand

29. What are the two types of CALL instructions? Discuss.

Ans. The two types are: intrasegment CALL and intersegment CALL.

If the operands are Near-proc, Memptr16 and Regptr16, then they specify intrasegment CALL while Far-proc and Memptr32 represent intersegment CALL.

30. Show the PUSH and POP instructions, as also the allowed operands.

Ans. The PUSH and POP instructions, as also the allowed operands are shown in Fig.14.16 (a) and (b) respectively.

Mnemonic	Meaning	Format	Operation	Flags affected
PUSH POP	Push word onto stack Pop word off stack	PUSH S POP D	((SP))←(S) (D)←((SP))	None None

(a)

Operand (S or D)
Register Seg-reg (CS illegal) Memory

(b)

Fig.14.16: (a) PUSH and POP instructions (b) Allowed operands

31. How many loop instructions are there and state their use.

Ans. There are in all three loop instructions. They can be used in place of certains conditional jump instructions. These instructions give the programmer a certain amount of flexibility in writing programs in a simpler manner.

32. List the different instructions and also the operations they perform.

Ans. The different loop instructions and the operations they perform are shown in Fig. 14.17.

33. What is meant by a 'string' and what are the characteristics of a string instruction?

Ans. A string is a series of data words (or bytes) that reside in successive memory locations. The characteristics of a string instruction are:

- Can move data from one block of memory locations to another one.
- A string of data elements stored in memory can be scanned for a specific data value. Successive elements of two strings can be compared to determine whether the two strings are same/different.

34. List the basic string instructions and the operations they perform.

Ans. The basic string instructions and the operations they perform are shown in Fig. 14.18.

Mnemonic	Meaning	Format	Operation
LOOP	Loop	LOOP Short-label	(CX)←(CX) − 1 Jump is initiated to location defined by short-label if (CX) ≠ 0; otherwise, execute next sequential instruction.
LOOPE/LOOPZ	Loop while equal/loop while zero	LOOPE/LOOPZ Short-label	(CX)←(CX) − 1 Jump to the location by short-label if (CX) ≠ 0 and (ZF) = 1; otherwise execute next sequential instruction.
LOOPNE/ LOOPNZ	Loop while not equal/ loop while not zero	LOOPNE/LOOPNZ Short-label	(CX)←(CX) − 1 Jump to the location defined by short label if (CX) ≠ 0 and (ZF) = 0; otherwise execute next sequential instruction

Fig. 14.17: LOOP instructions

Mnemonic	Meaning	Format	Operation	Flags Affected
MOVS	Move string	MOVS Operand	((ES)0+(DI))←((DS)0+(SI)) (SI)←(SI) ± 1 or 2 (DI)←(DI) ± 1 or 2	None
MOVSB	Move string byte	MOVSB	((ES)0+(DI))←((DS)0+(SI)) (SI)←(SI) ± 1 (DI)←(DI) ± 1	None

(Contd...)

MOVSW	Move string word	MOVSW	((ES)0+(DI))←((DS)0+(SI)) ((ES)0+(DI)+1)←((DS)0+(SI)+1) (SI)←(SI) ± 2 (DI)←(DI) ± 2	None
CMPS	Compare string	CMPS Operand	Set flags as per ((DS)0+(SI))–((ES)0+(DI)) (SI)←(SI) ± 1 or 2 (DI)←(DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
SCAS	Scan string	SCAS Operand	Set flags as per (AL or AX) – ((ES)0+(DI)) (DI)←(DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
LODS	Load string	LODS Operand	(AL or AX)←((DS)0+((SI)) (SI)←(SI) ± 1 or 2	None
STOS	Store string	STOS Operand	((ES)0+(DI))←(AL or AX)± 1 or 2 (DI)←(DI) ± 1 or 2	None

Fig.14.18: Basic string instructions

35. What is a 'REP' instruction? Discuss.

Ans. 'REP' stands for repeat and is used for repeating basic string operations—required for processing arrays of data.

There are a number of repeat instructions available and are used as a prefix in string instructions. The prefixes for use with the basic string instructions are shown in Fig. 14.19.

Prefix	Used with	Meaning
REP	MOVS STOS	Repeat while not end of string CX ≠ 0
REPE/REPZ	CMPS SCAS	Repeat while not end of string and strings are equal CX ≠ 0 and ZF = 1
REPNE/REPNZ	CMPS SCAS	Repeat while not end of string and strings are not equal CX ≠ 0 and ZF = 0

Fig.14.19: Prefixes for use with the basic string operations

36. Discuss the instructions for Autoindexing of string instructions.

Ans. When the system executes some string instruction, the addresses residing in DI and SI are incremented/decremented automatically. The content of direction flag (DF) decides the above. Two instructions—CLD (clear DF flag) or STD (Set DF flag) are used for the above and shown in Fig. 14.20.

Mnemonic	Meaning	Format	Operation	Flags affected
CLD	Clear DF	CLD	(DF)←0	DF
STD	Set DF	STD	(DF)←1	DF

Fig. 14.20: Instructions for autoincrementing and autodecrementing in string instructions

Execution of CLD (this makes DF = 0) permits autoincrement mode while execution of STD (this makes DF = 1) permits autodecrement mode.

SI or DI are autoincremented/autodecremented by one if a byte of data is processed or by two if a word of data is processed.

37. Write down the equivalent string instructions for the following two.

(i) MOV AL, [DI]	(ii) MOV AL, [SI]
CMP AL, [SI]	MOV [DI], AL
DEC SI	INC SI
DEC DI	INC DI

Ans. The following two are the equivalent string instructions of the given ones:

(i) STD	(ii) CLD
CMP SB	MOV SB

38. Where do memory source and destination addresses reside in string instructions?

Ans. The memory source and destination addresses in such cases are register SI in the data segment and DI in the extra segment.

15

Programming Techniques

1. Write an ALP (assembly language programming) for addition of two 8-bit data BB H and 11 H.

Ans.
```
0200  MOV AL, BB H          : 8-bit data BB H into AL
0202  MOV CL, 11 H          : 8-bit data 11 H into CL
0204  ADD AL, CL            : Contents of AL and CL added
0206  HLT                   : Stop.
```
Comment : Result in AL = CC H.

2. Write an ALP for addition of two 16-bit data BB11 H and 1122 H.

Ans.
```
0200  MOV AX, BB11 H        : 16-bit data BB11 H into AX
0203  MOV CX, 1122 H        : 16-bit data 1122 H into CX
0206  ADD AX, CX            : Contents of AX and CX added
0208  HLT                   : Stop
```
Comment : Result in AX = CC33 H.

3. Write an ALP for addition of two 8-bit data BB H and 11 H. The first data has an offset address of 0304 H and displacement.

Ans.
```
0200  MOV BX, 0304 H        : Offset address put in BX
0203  MOV AL, 11 H          : 8-bit data 11H into AL
0205  ADD AL, [BX + 07]     : 8-bit data from offset + displacement added with AL
0207  HLT                   : Stop.
```
Comment : Result in AL = CC H.

4. Write an ALP that subtracts 1234 H existing in DX from the word beginning at memory location MEMWDS.

Ans.
```
0200  MOV DX, 1234 H        : 16-bit data 1234 H put into DX
0203  SUB MEMWDS, DX        : Subtract data word 1234 H existing in DX from the data
                              word pointed to by MEMWDS.
0208  HLT                   : Stop.
```
Comment : If MEMWDS points to 3000 H then,

[3001 H : 3000 H] ← [3001 H : 3000 H] – 1234 H

5. Write an ALP which multiplies two 8-bit data 21 H and 17 H.

Ans.
```
0200  MOV AL, 21 H          : 8-bit multiplicand 21 H put into AL
0202  MOV CL, 17 H          : 8-bit multiplier 17 H put into CL
0204  MUL CL                : Contents of CL and AL are multiplied and the result
                              stored in AX
```

0206 HLT : Stop.
Comment : Result in AX = 02F7 H.

6. Write an ALP for dividing 1234 H by 34 H.

Ans.
0200 MOV AX, 1234 H : 16-bit dividend in 1234 H
0203 MOV CL, 34 H : 8-bit divisor in 34 H
0205 DIV CL : Content of AX divided by content of CL
0207 HLT : Stop.
Comment: Result in AX with
Quotient in AL = 59 H and
Remainder in AH = 20 H.

7. Write ALP that saves the contents of 8086's flags in memory location having an offset 1212 H and then to reload the flags from the contents of the memory location having an offset 2121 H.

Ans.
0200 LAHF : Load AH from flags
0201 MOV [1212], A H : Move the contents of AH to memory locations pointed to by offset 1212 H
0205 MOV AH, [2121] : Move the contents of memory locations pointed to offset 2121 H to AH
0209 SAHF : Store AH into flags
020A HLT : Stop.

8. Write an ALP that transfers a block of 100 bytes of data. The source and destination memory blocks start at 3000 H and 4000 H memory locations respectively. The data segment register value is DSADDR.

Ans.
2000 MOV AX, DSADDR : Move initial address of DS register into AX.
2003 MOV DS, AX : DS loaded with AX
2005 MOV SI, 3000 H : Source address put into SI.
2008 MOV DI, 4000 H : Destination address put into DI.
200B MOV CX, 64 H : Count value for number of bytes put into CX register
200D MOV AH, [SI] : Source byte moved into AH
200F MOV [DI], AH : AH byte moved into destination address
2011 INC SI : Increment source address
2012 INC DI : Increment destination address
2013 DEC CX : Decrement CX count
2014 JNZ 200D : Jump to 200D H until CX = 0
2017 HLT : Stop.

9. Write an ALP for ASCII addition of two numbers 2 H and 5 H.

Ans.
2000 MOV AL, 32 H : ASCII code 32 H for number 2 H is moved into AL
2002 MOV BL, 35 H : ASCII code 35 H for number 5 H is moved into BL
2004 AAA : ASCII adjust for addition
2005 HLT : Stop.
Result : (AL) = 07 H.

10. Write an ALP to find the average of two numbers.

Ans.
2000 MOV AL, 72 H : Get 1st number 72 H in AL
2002 ADD AL, 78 H : Add 2nd number 78 H with 72 H (in AL)
2004 ADC AH, 00 H : Put the carry in AH

2006 SAR AX, 1	: Divide Sum by 2
2008 MOV [3000 H], AL	: Copy AL content in memory location 3000 H
200B HLT	: Stop.

11. Write an ALP for moving a block consisting of 10 bytes from memory locations starting from 5000 H to memory locations starting from 6000 H. Use LOOP instruction.

Ans.

2000 CLD	: Clear direction flag
2001 MOV SI, 4000 H	: Source address put in SI
2004 MOV DI, 5000 H	: Destination address put in DI
2007 MOV CX, 000A H	: Put number of bytes to be transferred in CX.
200A MOV SB	: 1 byte copied from memory addressed by SI to addressed by DI.
200B LOOPNZ 200A H	: Loop till CX = 0
200D HLT	: Stop.

12. Write an ALP to find 2's complement of a string of 100 bytes.

Ans.

2000 CLD	: Clear direction flag
2001 MOV SI, 4000 H	: Source address put in SI
2004 MOV DI, 5000 H	: Destination address put in DI
2007 MOV CX, 0064 H	: Put the number of bytes to be 2's complemented in CX
200A LODSB	: Data byte to AL and INC SI
200B NEGAL	: 2's Complement of AL
200D STOSB	: Current AL value into DI and INC DI
200E LOOPNZ 200A H	: Loop till CX = 0.
2010 HLT	: Stop.

13. Write an ALP for block move of 100 bytes using Repeat instruction.

Ans.

2000 CLD	: Clear direction flag
2001 MOV SI, 4000 H	: Source address put in SI
2004 MOV DI, 5000 H	: Destination address put in DI
2007 MOV CX, 0064 H	: Put number of bytes to be block moved into CX
200A REPNZ	: Repeat till CX = 0
200B MOVSB	: Move data byte addressed by SI to DI.
2009 HLT	: Stop.

14. Write an ALP to evaluate X (Y + Z), where X = 10 H, Y = 20 H and Z = 30 H.

Ans.

2000 MOV AL, 20 H	: 20 H put in AL
2002 MOV CL, 30 H	: 30 H put in CL
2004 ADD AL, CL	: AL and CL are added up and result in AL
2006 MOV CL, AL	: AL transferred in CL
2008 MOV AL, 10 H	: 10 H put in AL
200A MUL CL	: AL and CL are multiplied and result in AL
200C MOV SI, 4000 H	: Source address in SI
200F MOV SI, AL	: AL put in SI
2011 HLT	: Stop.

15. Write an ALP that reverses the contents of the bytes TABLE through TABLE + N – 1.

Ans.

2000	MOV CL, N	: Number N put in CL
2002	MOV CH, 00 H	: 00 H put in CH
2004	MOV SI, TABLE	: Starting address of TABLE put in SI
2007	MOV DI, SI	: DI loaded with SI value
2009	SUB DI, 01	: 01 subtracted from DI
200B	ADD DI, CX	: CX value added to present DI value
200D	SHR CX, 01	: Divide CX count value by 2
200F	MOV AL, [SI]	: Move data pointed to by SI into AL
2011	XCHG AL, [DI]	: Exchange AL with data pointed to by DI
2013	MOV [SI], AL	: Save the exchanged data in SI
2014	INC SI	: Increment SI
2015	DEC DI	: Decrement DI
2016	LOOP 200F H	: Continue till CX = 0
2019	HLT	: Stop.

16. Write an ALP to find the maximum value of a byte from a string of bytes.

Ans.

2000	MOV SI, 3000 H	: Source address put in SI
2003	MOV CX, 0100 H	: Count value of bytes put in CX
2006	MOV AH, 00 H	: AH initialised with 00H
2008	CMP AH, [SI]	: AH compared with data pointed to by SI
200A	JAE 200E H	: Jump if AH is ≥ (SI) to 200E H
200C	MOV AH, [SI]	: Otherwise, move (SI) to AH
200E	INC SI	: Increment SI
200F	LOOPNZ 2008	: Loop unless CX ≠ 0
2011	MOV [SI], AH	: (AH) transferred to the memory location pointed to by SI
2013	HLT	: Stop.

17. Write an ALP to find the minimum value of a byte from a string of bytes.

Ans.

2000	MOV SI, 3000 H	: Source address put in SI
2003	MOV CX, 0100 H	: Count value of bytes put in CX
2006	MOV AH, 00 H	: AH initialised with 00H
2008	CMP AH, [SI]	: AH compared with data pointed to by SI
200A	JB 200E H	: Jump if (AH) < (SI) to 200E H
200C	MOV AH, [SI]	: Otherwise move (SI) to AH
200E	INC SI	: Increment SI
200F	LOOPNZ 2008	: Loop unless CX ≠ 0
2011	MOV [SI], AH	: (AH) transferred to the memory location pointed to by SI
2013	HLT	: Stop.

16

Modular Program Development and Assembler Directives

1. What is modular programming?

Ans. Instead of writing a large program in a single unit, it is better to write small programs—which are parts of the large program. Such small programs are called program modules or simply modules. Each such module can be separately written, tested and debugged. Once the debugging of the small programs is over, they can be linked together. Such methodology of developing a large program by linking the modules is called modular programming.

2. What are data coupling and control coupling?

Ans. Data coupling refers to how data/information are shared between two modules while control coupling refers to how the modules are entered and exited. Coupling depends on several factors like organisation of data as also whether the modules are assembled together or separately. The modular approach should be such that data coupling be minimized while control coupling is kept as simple as possible.

3. How modular programming helps assemblers?

Ans. Modular programming helps assembly language programming in the following ways :

- Use of macros-sections of code.
- Provide for procedures—i.e., subroutines.
- Helps data structuring such that the different modules can access them.

4. What is a procedure?

Ans. The procedure (or subroutine) is a set of codes that can be branched to and returned from.

The branch to a procedure is known as CALL and the return from the procedure is known as RETURN.

The RETURN is always made to the instruction just following the CALL, irrespective of where the CALL is located.

Procedures are instrumental to modular programming, although not all modules are procedures. Procedures have one disadvantage in that an extra code is needed to link them— normally referred to as linkage.

The CALL instruction pushes IP (and CS for a far call) onto the stack. When using procedures, one must remember that every CALL must have a RET. Near calls require near returns and far calls require far returns.

5. What are the two types of procedures?

Ans. There are two types of procedures. They are:

- Those that operate on the same set of data always.
- Those that operate on a new set of data each time they are called.

6. How are procedures delimited within the source code?

Ans. A procedure is delimited within a source code by placing a statement of the form < Procedure name > Proc < attribute > at the beginning of the procedure and the statement < Procedure name > ENDP at the end.

The procedure name acts as the identifier for calling the procedure and the attribute can be either NEAR or FAR—this attribute determines the type of RET statement.

7. Explain how a procedure and data from another module can be accessed.

Ans. A large program is generally divided into separate independent modules. The object codes for these modules are then linked together to generate a linked/executable file.

The assembly language directives: PUBLIC and EXTRN are used to enable the linker to access procedure and data from different modules. The PUBLIC directive lets the linker know that the variable/procedure can be accessed from other modules while the EXTRN directive lets the assembler know that the variable/procedure is not in the existing module but has to be accessed from another module. EXTRN directive also provides the linker with some added information about the procedure. For example,

```
EXTRN ROUTINE : FAR, TOKEN : BYTE
```

indicates to the linker that ROUTINE is a FAR procedure type and that TOKEN is a variable having type byte.

8. Discuss the technique of passing parameters to a procedure.

Ans. When calling a procedure, one or more parameters need to be passed to the procedure—an example being delay parameter. This parameter passing can be done by using one of the CPU registers like,

```
MOV CX, T
CALL DELAY
```

where, T represents delay parameter.

A second technique is to use a memory location like,

```
MOV TEMP, T
CALL DELAY
```

where, TEMP is representative of memory locations.

A third technique is to pass the address of the memory variable like,

```
MOV SI, POINTER
CALL DELAY
```

while in the procedure, it extracts the delay parameter by using the instruction MOV CX, [SI].

This way an entire table of values can be passed to a procedure. The above technique has the inherent disadvantage of a register or memory location being dedicated to hold the parameter when the procedure is called. This problem becomes more prominent when using nested procedures. One alternative is to use the stack to relieve registers/memory locations being dedicated like,

```
MOV CX, T
PUSH CX
CALL DELAY
```

The procedure then can pop off the parameters, when needed.

9. Explain the term Assembler Directive.

Ans. There are certain instructions in the assembly language program which are not a part of the instruction set. These special instructions are instructions to the assembler, linker and loader and control the manner in which a program assembles and lists itself. They come into play during the assembly of a program but do not generate any executable machine code.

As such these special instructions—which, as told, are not a part of the instruction set —are called assembler directives or pseudo-operations.

10. Give a tabular form of assembler directives.

Ans. Table 16.1 gives a summary of assembler directives.

Table 16.1: Summary of assembler directives

Directive	Action
ALIGN	aligns next variable or instruction to byte which is multiple of operand
ASSUME	selects segment register(s) to be the default for all symbol in segment(s)
COMMENT	indicates a comment
DB	allocates and optionally initializes bytes of storage
DW	allocates and optionally initializes words of storage
DD	allocates and optionally initializes doublewords of storage
DQ	allocates and optionally initializes quadwords of storage
DT	allocates and optionally initializes 10-byte-long storage units
END	terminates assembly; optionally indicates program entry point
ENDM	terminates a macro definition
ENDP	marks end of procedure definition
ENDS	marks end of segment or structure
EQU	assigns expression to name
EVEN	aligns next variable or instruction to even byte
EXITM	terminates macro expansion
EXTRN	indicates externally defined symbols
LABEL	creates a new label with specified type and current location counter
LOCAL	declares local variables in macro definition
MACRO	starts macro definition
MODEL	specifies mode for assembling the program

11. Explain the following assembler directives (a) CODE (b) ASSUME (c) ALIGN

Ans. (a) CODE

It provides a shortcut in the definition of the code segment. The format is Code [name]

Here, the 'name' is not mandatory but is used to distinguish between different code segments where multiple type code segments are needed in a program.

(b) ASSUME

The four physical segments viz., CS, DS, SS and ES can be directly accessed by 8086 at any given point of time. Again 8086 may contain a number of logical segments which can be assigned as physical segments by the ASSUME directive. For example

ASSUME CS: Code, DS : Data, SS : Stack

(c) ALIGN

This directive forces the assembler to align the next segment to an address that is divisible by the number that follows the ALIGN directive. The general format is

ALIGN number

where number = 2, 4, 8, 16

ALIGN 4 forces the assembler to align the next segment at an address that is divisible by 4. The assembler fills the unused byte with 0 for data and with NOP for code.

Normally, ALIGN 2 is used to start a data segment on a word boundary while ALIGN 4 is used to start a data segment on a double word boundary.

12. Explain the DATA directive.

Ans. It is a shortcut definition to data segments. The directives DB, DW, DD, DR and DT are used to (a) define different types of variables or (b) to set aside one or more storage locations in memory-depending on the data type

DB — Define Byte
DW — Define Word
DD — Define Double word
DQ — Define Quadword
DT — Define Ten Bytes

ALPHA DB, 10 H, 16 H, 24 H; Declare array if 3 bytes names; ALPHA

13. Explain the following assembler directives : (a) DUP (b) END (c) EVEN

Ans. **(a) DUP:** The directive is used to initialise several locations and to assign values to these locations. Its format is: Name Data-Type Num DUP (value)

As an Example:

```
TABLE DOB 20 DUP(0)  ;  Reserve an array of 20
                     ;  bytes of memory and initialise all 20
                     ;  bytes with 0. Array is named TABLE
```

(b) END: This directive is put in the last line of a program and indicates the assembler that this is the end of a program module. Statement, if any, put after the END directive is ignored. A carriage return is obviously required after the END directive.

(c) EVEN: This directive instructs the assembler to advance its location counter in such a manner that the next defined data item or label is aligned on an even storage boundary. It is very effectively used to access 16 or 32-bits at a time. As an example.

```
EVEN LOOKUP DW 10 DUP (0)   ; Declares the array of 10 words
                            ; starting from an even address.
```

14. Discuss the MODEL directive.

Ans. This directive selects a particular standard memory model. Each memory model is characterised by having a maximum space with regard to availability of code and data. This different models are distinguished by the manner by which subroutines and data are reached by programs.

Table 16.2 gives an idea about the different models with regard to availability of code and data.

Table16.2: The different models

Model	Code segments	Data segments
Small	One	One
Medium	Multiple	One
Compact	One	Multiple
Large	Multiple	Multiple

15. Give a typical program format using assembler directives.

Ans. A typical program format using assembler directives is as shown below:

```
Line 1.   MODEL SMALL   ;  selects small model
Line 2.   DATA          ;  indicates data segment
          ...
          ...
          ...
Line 15.  CODE          ;  indicates start of code segment
          ...
          ...              body of the program
          ...
Line 20.  END           ;  End of file
```

16. Discuss the PTR directive.

Ans. This directive assigns a specific type to a variable or a label and is used in situations where the type of the operand is not clear. The following examples will help explain the PTR directive more elaborately.

(a) The instruction INC [BX] does not tell the assembler whether to increment a byte or word pointed to by BX. This ambiguity is cleared with PTR directive.
INC BYTE PTR [BX] ; Increment the byte pointed to by [BX]
INC WORD [BX] ; Increment the word pointed to by [BX]

(b) An array of words can be accessed by the statement WORDS, as for examples
WORDS DW 1234 H, 8823 H, 12345 H, etc.
But PTR directive helps accessing a byte in an array, like,
MOV AH, BYTE PTR WORDS.

(c) PTR directive finds usage in indirect jump. For an instruction like JMP [BX], the assembler cannot decide of whether to code the instruction for a NEAR or FAR jump. This difficulty is overcome by PTR directive.
JMP WORD PTR [BX] and JMP DWORD PTR [BX] are examples of NEAR jump and FAR jump respectively.

17. What is a macro?

Ans. A macro, like a procedure, is a group of instructions that perform one task. The macro instructions are placed in the program by the macro assembler at the point it is invoked.

Use of macros helps in creating new instructions that will be recognised by the assembler. In fact libraries of macros can either be written or purchased and included in the source code which apparently expands the basic instruction set of 8086.

18. Show the general format of macros.

Ans. The general format of a macro is

NAME MACRO Arg 1 Arg 2 Arg 3

Statements ....

....

ENDM

The format begins with NAME which is actually the name assigned to the MACRO. 'Arg's' represent the arguments of the macro. Arguments are optional in nature and allows the same macro to be used in different places within a program with different sets of data. Each of the arguments represent a particular constant, hence a CPU register, for instance, cannot be used.

All macros end with ENDM.

19. Explain macro definition, macro call and macro expansion.

Ans. Creation of macro involves insertion of a new opcode that can be used in the program. This code, often called prototype code, along with the statements for representing and terminating a macro is called macro definition.

The statements that follow a macro definition is called macro call.

When the assembler encounters a macro call, it replaces the call with macro's code. This replacement action is referred to as macro expansion.

20. Explain the INCLUDE file.

Ans. A special file, say MACRO.LIB can be created which would contain the definitions of all macros of the user. In such a case, the writing of each macro's definition at the head of the main program can be dispensed with. The INCLUDE file may look like.

INCLUDE MACRO.LIB

This statement forces the assembler to automatically include all the statements in the MACRO.LIB.

Sometimes it may be undesirable to include the INCLUDE statement when the INCLUDE file is very long and the user may not be using many of the macros in the file.

21. Explain local variables in a macro.

Ans. Within the body of a macro, local variables can be used. A local variable can be defined by using LOCAL directive and is available within the macro and not outside.

For example, a local variable can be used in a jump address. The jump address has to be defined as a local, an error message will be outputted by the assembler.

Local variable(s) must be defined immediately following the macro directive, with the help of local directives.

22. Explain Controlled Expansion (also called Conditional Assembly).

Ans. While inside the macro, facilities are available to either accept or reject a code during macro execution— i.e., expansion of a macro prototype code would depend on the type(s) of actual parameter(s) passed to it by the call. This facility of selecting a code that is to be assembled is called controlled expansion.

The conditional assembly statements in macro are:

IF-ELSE-ENDIF	Statement
REPEAT	Statement
WHILE	Statement
FOR	Statement

23. For the conditional assembly process, show the (a) forms used for the IF statement (b) relational operators used with WHILE and REPEAT.

Ans. Figure 16.1 and 16.2 show respectively the forms used for the IF statement and the relational operators used with WHILE and REPEAT.

Statement	Function
IF	If the expression is true
IFB	If argument is blank
IFE	If the expression is not true
OFDEF	If the label has been defined
IFNB	If argument is not blank
IFNDEF	If the label has not been defined
IFIDN	If argument 1 equals argument 2
IFDIFWWW	If argument 1 does not equal argument 2

Fig.16.1: Forms used for IF statements

Operator	Function
EQ	Equal
NE	Not Equal
LE	Less than or Equal
LT	Less than
GT	Greater than
GE	Greater than or Equal
NOT	Logical inversion
AND	Logical AND
OR	Logical OR
XOR	Logical XOR

Fig.16.2: Relational operators used with WHILE and REPEAT

24. Distinguish between macro and procedure.

Ans. A procedure is invoked with a CALL instruction and terminated with a RET instruction. Again the code for the procedure appears only once in the programs—irrespective of the number of times it appears.

A macro is invoked, on the other hand, during program assembly and not when the program is run. Whenever in the program the macro is required, assembler substitutes the defined sequence of instructions corresponding to the macro. Hence macro, if used quite a few number of times, would consume a lot of memory space than that would be required by procedure.

Macro does not require CALL–RET instructions and hence will be executed faster. Sometimes, depending on the macro size, the macro may require less number of codes than is required by the equivalent procedure.

17

Input/Output Interface of 8086

1. What are the two schemes employed for I/O addressing.

Ans. The two schemes employed for I/O addressing are Isolated I/O and Memory I/O.

2. Compare Isolated I/O and Memory mapped I/O.

Ans. The comparison is shown in Table 17.1

Table 17.1: Comparison between isolated and memory mapped I/O

Isolated I/O	Memory mapped I/O
1. I/O devices are treated separate from memory.	1. I/O devices are treated as part of memory.
2. Full 1 MB address space is available for use as memory.	2. Full 1 MB cannot be used as memory since I/O devices are treated as part of memory.
3. Separate instructions are provided in the instruction set to perform isolated I/O input-output operations. These maximise I/O operations.	3. No separate instructions are needed in this case to perform memory mapped I/O operations. Hence, the advantage is that many instructions and addressing modes are available for I/O operations.
4. Data transfer takes place between I/O port and AL or AX register only. This is certainly a disadvantage.	4. No such restriction in this case. Data transfer can take place between I/O port and any internal register. Here, the disadvantage is that it somewhat slows the I/O operations.

3. Draw the Isolated I/O memory and I/O address space.

Ans. In isolated I/O scheme, memory and I/O are treated separately. The 1 MB address space can be treated as memory address space ranging from 00000 H to FFFFF H, while the address range from 00000 H to 0FFFF H (i.e., 64 KB I/O addresses) can be treated as I/O address space as shown below in Fig. 17.1.

It should be remembered that two consecutive memory or I/O addresses could be accessed as a word-wide data.

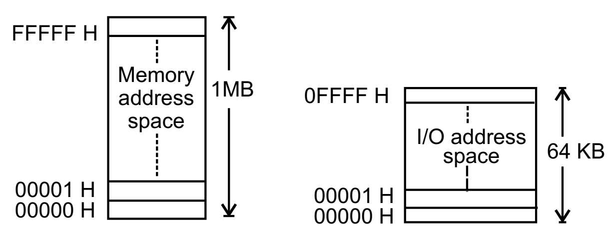


Fig. 17.1: Isolated I/O (a) memory address space (b) I/O address space

4. Draw and explain the memory mapped I/O scheme for 8086.

Ans. In this scheme, CPU looks to I/O ports as if it is part of memory. Some of the memory space is earmarked (dedicated) for I/O ports or addresses. The memory mapped I/O scheme is shown in Fig. 17.2 below in which the memory locations starting from C0000 H to C0FFF H (4 KB in all) and from D0000 H to D0FFF H (4 KB in all) are assigned to I/O devices.

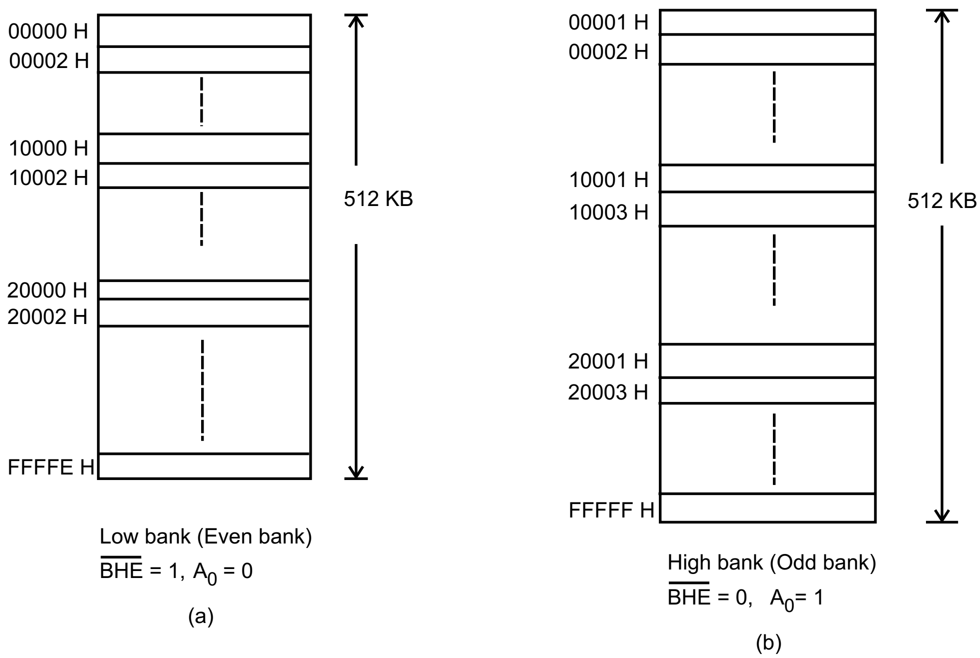


Fig. 17.2 : Memory mapped I/O scheme

5. Draw the (a) MIN and (b) MAX mode 8086 based I/O interface.

Ans. **(a)** The 8086 based Min mode I/O interface is shown below in Fig. 17.3.

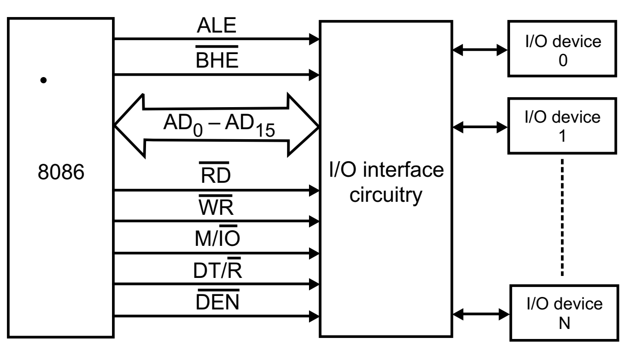


Fig. 17.3: The Minimum mode 8086 system I/O interface

It is seen that the lower two bytes $AD_0 - AD_{15}$ are used for input/output data transfers. The interface circuitry performs the following tasks.

- Selecting the particular I/O port
- Synchronise data transfer
- Latch the output data
- Sample the input data
- Voltage levels between the I/O devices and 8086 are made compatible.

(b) The 8086 based Max mode I/O interface is shown below in Fig.17.4

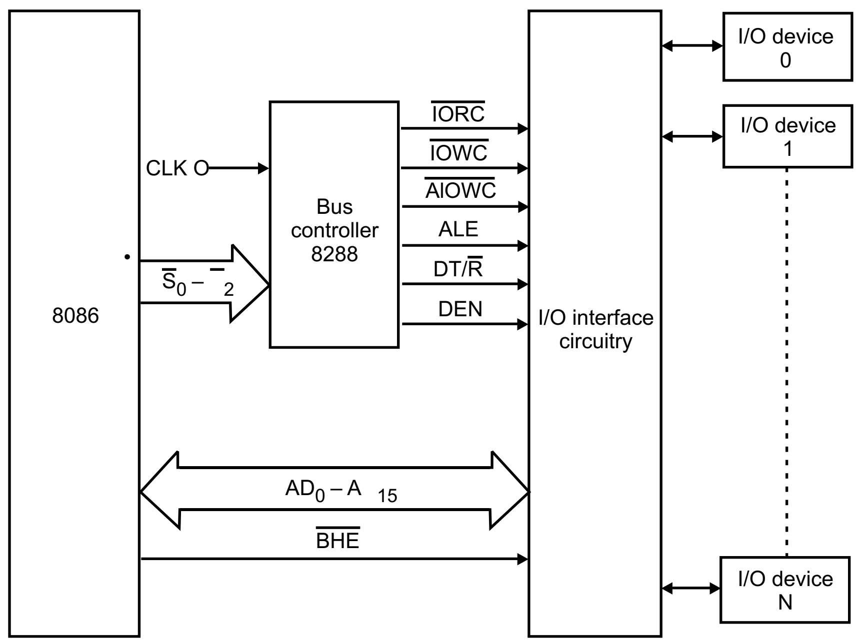


Fig. 17.4: The Maximum-mode 8086 system I/O interface

In this case the status codes $\overline{S_2} - \overline{S_0}$ outputted by 8086 are fed to the 8288 bus controller IC. The decoder circuit within 8288 decodes these three signals. For instance, 001 and 010 on $\overline{S_2}\ \overline{S_1}\ \overline{S_0}$ lines indicate 'Read I/O port' and 'Write I/O port' respectively. The first corresponds to $\overline{IORC}$ while the second corresponds to $\overline{IOWC}$ or $\overline{AIOWC}$ signals. These command signals are utilised to control data flow and its direction between I/O devices and the data bus.

6. What kind of I/O is used for IN and OUT instructions?

Ans. For 8086 based systems, isolated I/O is used with IN and OUT instructions. The IN and OUT instructions are of two types: direct I/O instructions and variable I/O instructions. The different types of instructions are tabulated in Fig.17.5.

Mnemonic	Meaning	Format	Operation
IN	Input direct	IN Acc, Port	(Acc)←(Port) Acc = AL or AX
	Input indirect (variable)	IN Acc, DX	(Acc)←((DX))
OUT	Output direct	OUT Port, Acc	(Port)←(Acc)
	Output indirect (variable)	OUT DX, Acc	((DX))←(Acc)

Fig. 17.5: Input/output instructions

7. Which register(s) is/are involved in data transfers?

Ans. Only AL (for 8-bits) or AX (for16-bits) register is involved in data transfer involving the 8086's CPU and I/O devices—thus they are also known as accumulator I/O.

8. Bring out the differences between direct I/O instructions and variable I/O instructions.

Ans. The differences between the two types of instructions are tabulated below in Table 17.2.

Table 17.2: Differences between direct and variable I/O instructions

Direct I/O	Variable I/O
1. Involves 8-bit address as part of instruction	1. Involves 16-bit address as part of instruction. This resides in DX register. It must be borne in mind that the value in DX register is not an offset, but the actual port address.
2. Can access a maximum of 2^8 = 255 byte addresses.	2. Can access a maximum of 2^{16} = 64 KB of addresses.

9. Give one example each of (a) direct I/O (b) variable I/O instruction.

Ans. (a) An example of direct I/O instruction is as follows:

IN AL, 0F2 H

On execution, the contents of the byte wide I/O port at address location F2 H will be put into AL register.

(b) An example of this type is:

MOV DX, 0C00F H
IN AL, DX

On execution, at first DX register is loaded with the input port having address C00F H. The second instruction ensures that the port content is moved over to AL register.

10. Draw the (a) in port and (b) out port bus cycle of 8086.

Ans. (a) The input bus cycle of 8086 is shown in the Fig.17.6. In the first T state (i.e., T_1), address comes out via A_0–A_{19}, along with $\overline{BHE}$ signal. Also ALE signal goes high in T_1. The high to low transition on ALE at the end of T_1 latches the address bus. $M/\overline{IO}$ signal goes low at the beginning of T_1. $\overline{RD}$ line goes low in T_2 while data transfer occurs in T_3. DT/$\overline{R}$ goes low at the beginning of T_1 and $\overline{DEN}$ signal becomes active in T_2 which tells the I/O interface circuit when to put data on the data bus.

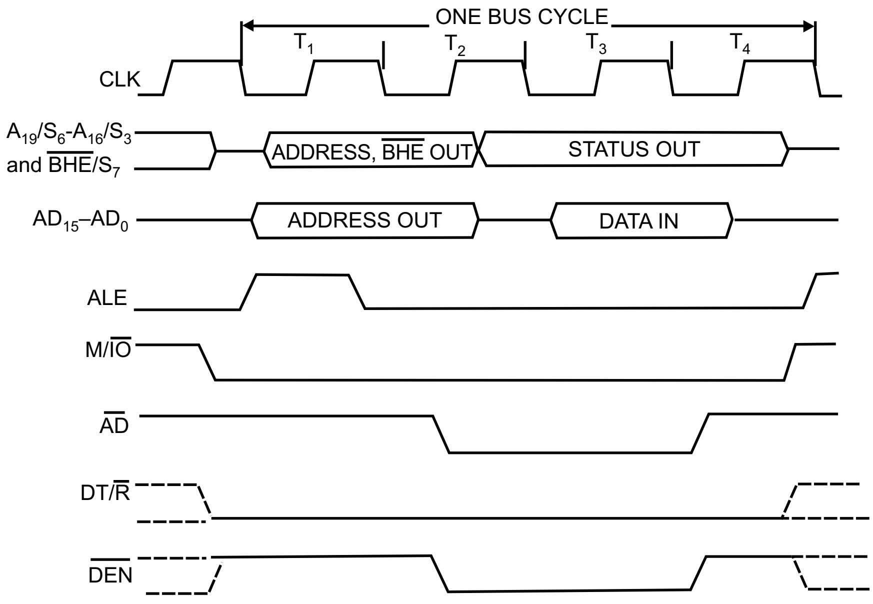


Fig. 17.6 : The Input bus cycle of 8086

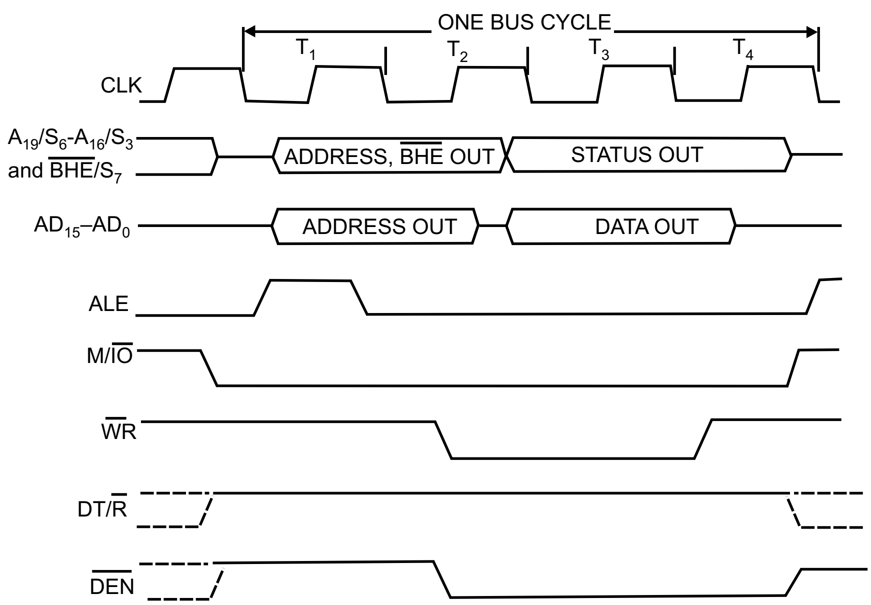


Fig. 17.7: The Output bus cycle of 8086

(b) The output bus cycle shown in Fig. 17.7.

The main differences between the output bus cycle and the just discussed input bus cycle are:

- $\overline{WR}$ Signal becomes active earlier than the $\overline{RD}$ signal. Hence in this case valid data is put on the data bus in T_2 state.
- $\overline{DEN}$ signal becomes active in T_1, while the same occurs in T_2 state for input bus cycle.

18

8086 Interrupts

1. How many interrupts can be implemented using 8086 µP?

Ans. A total of 256 interrupts can be implemented using 8086 µP.

2. Mention and tabulate the different types of interrupts that 8086 can implement.

Ans. 8086 µP can implement seven different types of interrupts.

- NMI and INTR are external interrupts implemented via *Hardware*.
- INT n, INTO and INT3 (breakpoint instruction) are software interrupts implemented through *Program*.
- The 'divide-by-0' and 'Single-step' are interrupts *initiated by CPU*.

Table 18.1 shows the seven interrupt types implemented by 8086.

Table 18.1: The Seven different types of 8086 interrupts

Name	Initiated by:	Maskable?	Trigger	Priority	Acknowledge signal?	Vector table address[a]	Interrupt latency
NMI	External hardware	No	↑Edge, hold 2 T states min.	2	None	00008H–0000BH	Current instruction + 51 T states
INTR	External hardware	Yes via IF	High level until acknowledged	3	$\overline{\text{INTA}}$	n * 4[b]	Current instruction + 61 T states
INT n	Internal via software	No	None	1	None	n * 4	51 T states
INT 3 (break point)	Internal via software	No	None	1	None	0000CH–0000FH	52 T states
INTO	Internal via software	No	None	1	None	00010H–00013H	53 T states
Divide-by-0	Internal via CPU	Yes via OF	None	1	None	00000H–00003H	51 T states
Single-step	Internal via CPU	Yes via TF	None	4	None	00004H–00007H	51 T states

a. All interrupt types cause the flags, CS, and IP registers to be pushed onto the stack. In addition, the IF and TF flags are cleared.

b. n is an 8-bit type number read during the second INTA pulse.

3. Distinguish between the two hardware interrupts of 8086.

Ans. The distinction between the two hardware interrupts of 8086 are as follows, shown in Table 18.2.

Table18.2: Comparison of NMI and INTR interrupts

NMI	INTR
1. Non-maskable type.	1. Maskable type.
2. Higher priority.	2. Lower priority.
3. Edge triggered interrupt initiated on Low to High transition.	3. Level triggered interrupt.
4. Must remain high for more than 2 CLK cycles.	4. Sampled during last CLK cycle of each instruction.
5. The rising edge of NMI input is latched on-chip and is serviced at the end of current instruction.	5. No latching. Must stay high until acknowledged by CPU.
6. No acknowledgement.	6. Acknowledged by INTA output signal.

4. How many bytes are needed to store the starting addresses of ISS for 8086 μP?

Ans. 8086 μP can implement 256 different interrupts. To store the starting address of a single ISS (Interrupt Service Subroutine), four bytes of memory space are required—two bytes to store the value of CS and two bytes to store the IP value. Thus to store the starting address of 256 ISS, in all 256 × 4 = 1024 bytes = 1 KB will be required.

5. Indicate the number of memory spaces needed in stack when an interrupt occurs.

Ans. When an interrupt occurs, before moving over to starting address of the corresponding ISS, the following are pushed into the stack: the contents of the flag register, CS and IP. Since each one of the three are 2 bytes, hence a total of 6 bytes of memory space is needed in the stack to accommodate the flag register, CS and IP contents.

6. What are meant by interrupt pointer and interrupt pointer table?

Ans. The starting address of an ISS in the 1 KB memory space is known as the interrupt pointer or interrupt vector corresponding to that interrupt.

The 1 KB memory space needed to store the starting addresses of all the 256 ISS is called the interrupt pointer table.

7. Write down the steps, sequentially carried out by the systems when an interrupt occurs.

Ans. When an interrupt occurs (hardware or software), the following things happen:

- The contents of flags register, CS and IP are pushed on to the stack.
- TF and IF are cleared which disable single step and INTR interrupts respectively.
- Program jumps to the starting address of ISS.
- At the end of ISS, when IRET is executed in the last line, the contents of flag register, CS and IP are popped out of the stack and placed in the respective registers.
- When the flags are restored, IF and TF get back their previous values.

8. Draw and discuss the interrupt pointer table for 8086 μP.

Ans. The interrupt pointer table for 8086 is shown in Fig. 18.1.

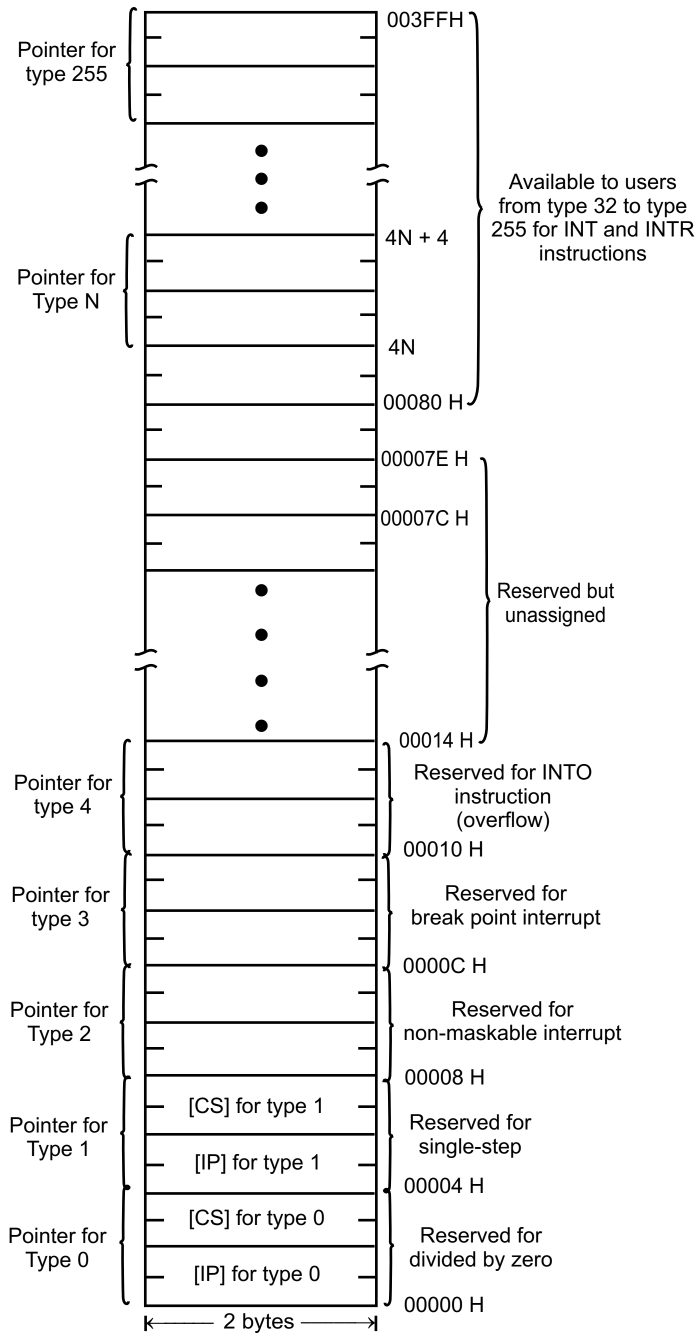


Fig. 18.1: Interrupt Pointer Table for intel 8086

The 256 interrupt pointers are stored in memory locations starting from 00000 H to 003FF H (1 KB memory space). The number assigned to an interrupt pointer is called

the Type of the corresponding interrupt. As for example, Type 0 interrupt, Type 1 interrupt ... Type 255 interrupt. Type 0 interrupt has a memory address 00000 H, Type 1 has a memory address 00004 H, while Type 255 has a memory address 003FF H. The first five pointers (Type 0 to Type 4) are dedicated pointers used for divide by zero, single step, NMI, break point and overflow interrupts respectively. The next 27 pointers (Type 5 to Type 31) are reserved pointers—reserved for some special interrupts. The remaining 224 interrupts—from Type 32 to Type 255 are available to the programmer for handling hardware and software interrupts.

9. Discuss the priority of interrupts of 8086.

Ans. 8086 tests for the occurence of interrupts in the following hierarchical sequence:

- Internal interrupts (divide-by-0, single step, break point and overflow)
- Non-maskable interrupt—via NMI
- Software interrupts—via INTn
- External hardware interrupt—via INTR

Hence, internal interrupts belong to the highest priority group and internal hardware interrupts are the lowest priority group. Again, different interrupts are given different priorities by assigning a type number corresponding to each priority—starting from Type 0 (highest priority interrupt) to Type 255 (lowest priority interrupt). Thus, Type 40 interrupt is having more priority than Type 41 interrupt. If we presume that at any instant a Type 40 interrupt is in progress, then it can be interrupted by any software interrupt, the non-maskable interrupt, all internal interrupts or any external interrupt with a Type number less than 40.

10. Outline the events that take place when 8086 processes an interrupt.

Ans. Fig. 18.2 shows the manner in which 8086 processes an interrupt while the following are the events that take place sequentially when the processor receives an interrupt (from an *external* device via INT 32 through INT 255:

- Receiving an interrupt request (from an external device)
- Generation of interrupt acknowledge bus cycles.
- Servicing the Interrupt Service Subroutine (corresponding to the external device which has interrupted the CPU.)

Again the difference between simultaneous interrupt and interrupt within an ISS is to be understood. Occurrence of more than one interrupt within the *same instruction* is called simultaneous interrupt while if an interrupt occurs while the ISS is in progress, it is an interrupt occurring within an ISS.

Internal interrupts (except single step) have priority over simultaneous external requests. For example, let the current instruction causes a divide-by-zero interrupt when an INTR (a hardware interrupt) occurs, the former will be serviced. Again, if simultaneous interrupts occur on INTR and NMI, then NMI will be serviced first.

For simultaneous interrupts occurring, the priority structure of Fig.18.2 will be honoured, with the highest priority interrupt being serviced first.

Although it has been commented that software interrupts get priority over external hardware interrupts, even then if an interrupt on NMI occurs as soon as the interrupt's ISS begins, it (i.e., NMI) will be recognised and hence serviced.

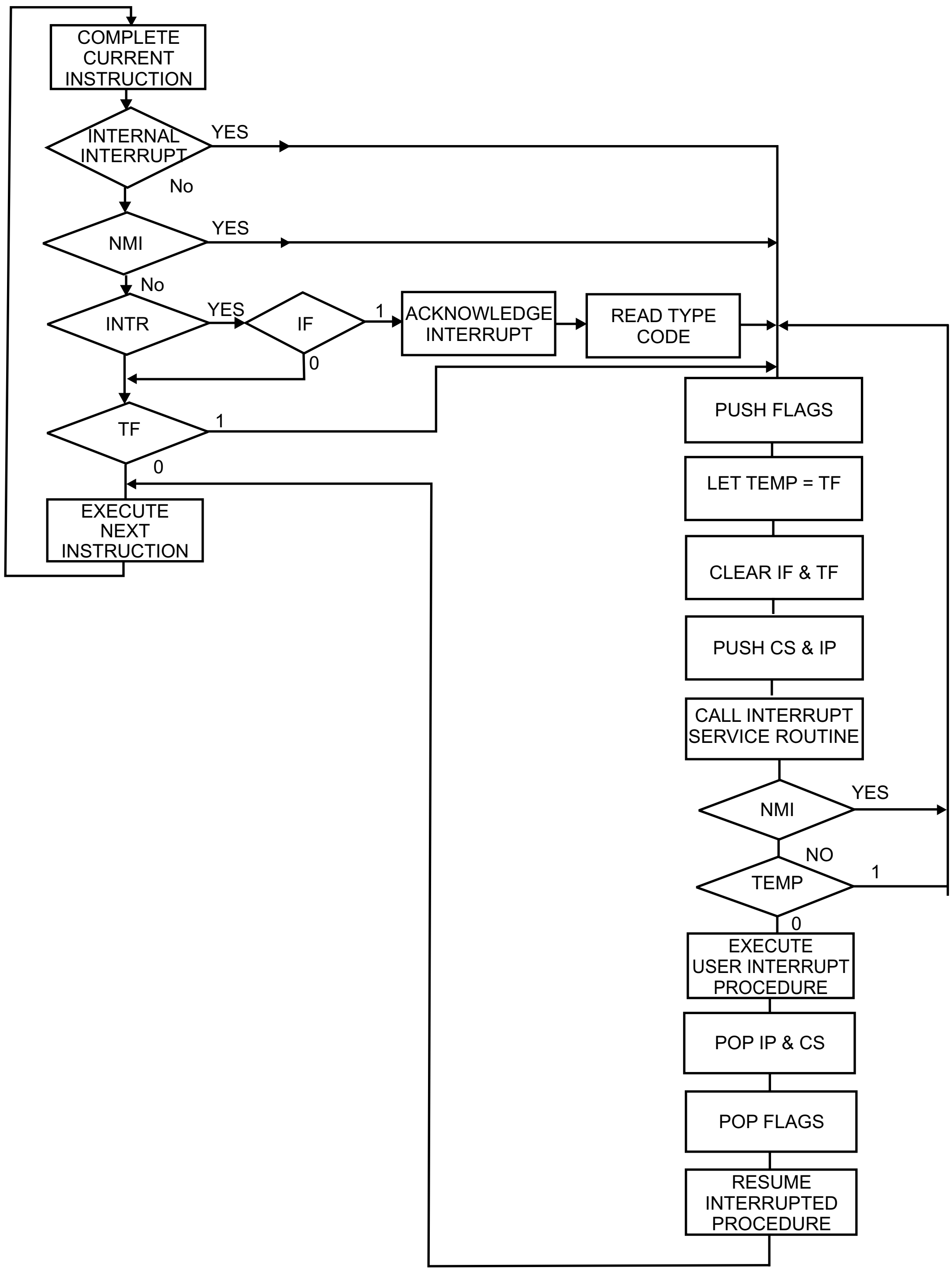


Fig.18.2: Interrupt processing sequence of the 8086 microprocessors

11. List the different interrupt instructions associated with 8086 µP.

Ans. Table 18.3 lists the different interrupts of 8086 µP along with a brief description of their functions.

Table18.3 : The different interrupt instructions of 8086 µP

Mnemonic	Meaning	Format	Operation	Flags affected
CLI	Clear interrupt flag	CLI	0→(IF)	IF
STI	Set interrupt flag	STI	1→(IF)	IF
INT n	Type n software interrupt	INT n	(Flags)→((SP) − 2) 0→TF, IF. (CS)→((SP) − 4) (2 + 4, n)→(CS) (IP)→((SP)–6) (4 . n)→(IP)	TF, IF
IRET	Interrupt return	IRET	((SP))→(IP) ((SP)+2)→(CS) ((SP)+4)→(Flags) (SP)+6→(SP)	All
INTO	Interrupt on overflow	INTO	INT 4 steps	TF, IF
HLT	Halt	HLT	Wait for an external interrupt or reset to occur	None
WAIT	Wait	WAIT	Wait for $\overline{\text{TEST}}$ input to go active	None

12. Show the internal interrupts and their priorities.

Ans. The internal interrupts are: Divide-by-0, single step, break point and overflow corresponding to Type 0, Type 1, Type 3 and Type 4 interrupts respectively.

Since a type with lesser number has higher priority than a type with more number, thus the mentioned internal interrupts can be arranged in a decreasing priority mode, with highest priority mentioned first: Divide-by-0, single step, break point, overflow.

13. What are the characteristics associated with internal interrupts?

Ans. The following are the characteristics associated with internal interrupts:

- The interrupt type code is either contained in the instruction itself or is predefined.
- No $\overline{\text{INTA}}$ bus cycles are generated, as in the case of INTR interrupt input.
- Apart from single step interrupt, no other internal interrupt can be disabled.
- Internal interrupts, except single step have higher priority than external interrupts.

14. Discuss the two interrupts HLT and WAIT.

Ans. On execution of HLT (halt) instruction by 8086, CPU suspends its instruction execution and enters into an idle state. It waits for either an external hardware interrupt or a reset input (interrupt). When any one of these occurs, CPU starts executing again.

When the WAIT instruction is executed by 8086, it internally checks the logic level existing at its $\overline{\text{TEST}}$ input. If $\overline{\text{TEST}}$ is at logic 1 state, then CPU goes into an idle state. When $\overline{\text{TEST}}$ input assumes a zero state, execution resumes from the next sequential

instruction in the program. $\overline{\text{TEST}}$ input is normally connected to the BUSY output signal of 8087 NDP.

15. Mention the addresses at which CS_{40} and IP_{40} corresponding to vector 40 would be stored in memory.

Ans. INT 40, for its storage, requires four memory locations—two for IP_{40} and two for CS_{40}.

The addresses are calculated as follows:

$$4 \times 40 = 160_{10} = 1010\ 0000_2 = \text{A0 H}.$$

Thus, IP_{40} is stored starting at 000A0 H and CS_{40} is stored starting at 000A2 H.

16. Explain in detail the external hardware interrupt sequence.

Ans. The external device can request for service via INT 32 through INT 255 by pulling the corresponding INT n (n = 32 to 255) high. The interrupt request gives rise to generation of interrupt acknowledge bus cycles and then moving into ISS corresponding to the device which has interrupted the system. The presently requested interrupt is recognised provided no higher priority interrupt is pending and IF is already set via software.

Once the interrupt is recognised (since for any INTR to be recognised, the corresponding INT must stay high till the last clock cycle of the presently executed instruction). 8086 initiates interrupt acknowledge bus cycles, shown in Fig. 18.3.

During T_1 of the first interrupt bus cycle, ALE is put to low state and remains so till the end of the cycle. During the whole of this cycle, address/data bus is driven into Z-state. During T_2 and T_3 of this first interrupt bus cycle, $\overline{\text{INTA}}$ is put to low state—indicating that the request for service has been granted so that the requesting device can withdraw its high logic which is connected to INTR pin 8086.

$\overline{\text{LOCK}}$ signal is of importance only in maximum mode. This signal goes low during T_2 of the first INTA bus cycle and is maintained in zero state until T_2 of the second INTA bus cycle.

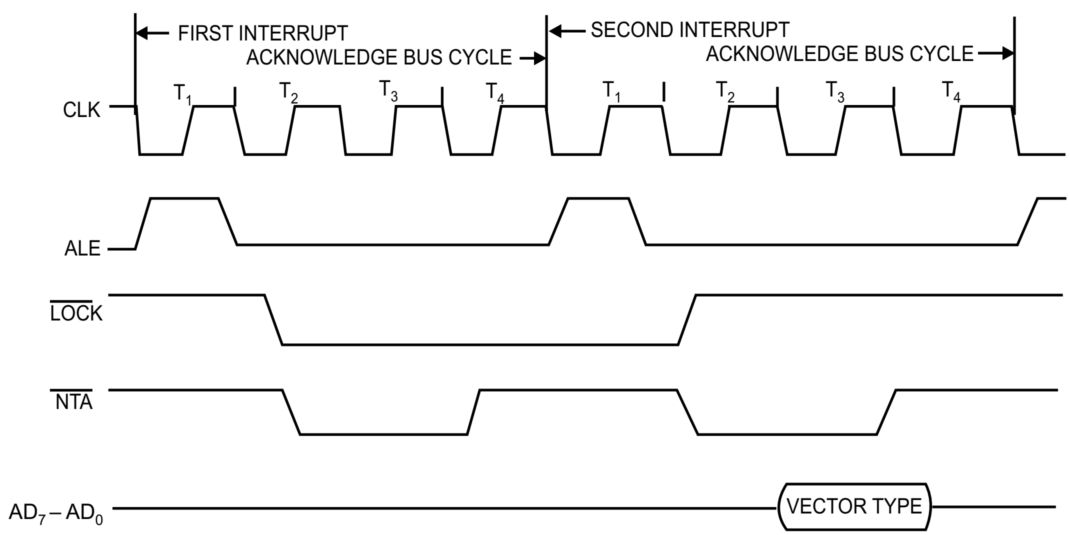


Fig. 18.3: Interrupt acknowledge bus cycle

8086 is prevented from accepting a HOLD request. The $\overline{\text{LOCK}}$ output, in conjunction with external logic, is used to lock off other devices from the system bus. This ensures completion of the current interrupt till its completion.

During the second interrupt bus cycle, the external circuit puts in the interrupt code (32_{10} = 20 H through 255_{10} = FF H) on the data bus AD_0– AD_7 during T_3 and T_4 and is read by 8086.

Before moving to ISS, CPU saves the contents of the flag register along with the current CS and IP values. Now by reading the number from the data bus, the corresponding CS and IP values are placed in them (for instance, if the external device has interrupted via INT 60, then CS_{60} and IP_{60} would be loaded into CS and IP register respectively). Thus the ISS would be run to its completion because before moving into ISS, IF and single-stepping have been disabled.

In the last line of ISR, an IRET instruction is there, which on its execution pops the old CS and IP values from the stack and put them in CS and IP registers. This thus ensures that the main program starts at the very memory location that was left off because of ISS.

17. Indicate two applications where NMI interrupt can be applied.

Ans. NMI is a non-maskable hardware interrupt, i.e., it cannot be masked or disabled. Hence, it is used for very important system exigencies like (a) detection of power failure or (b) detection of memory read error cases.

18. Draw a circuit that will terminate the INTR when interrupt request has been acknowledged.

Ans. Fig. 18.4 makes INTR input of 8086 to go into 1 state once the interrupt request comes from some external agency. The falling edge of the peripheral clocks the flip-flop which makes INTR to become 1. The first $\overline{\text{INTR}}$ pulse then resets Q, making INTR to become 0. This ensures that no second interrupt request is recognised by the system. The reset input sees to it that INTR remains in the 0 state when the system is reset.

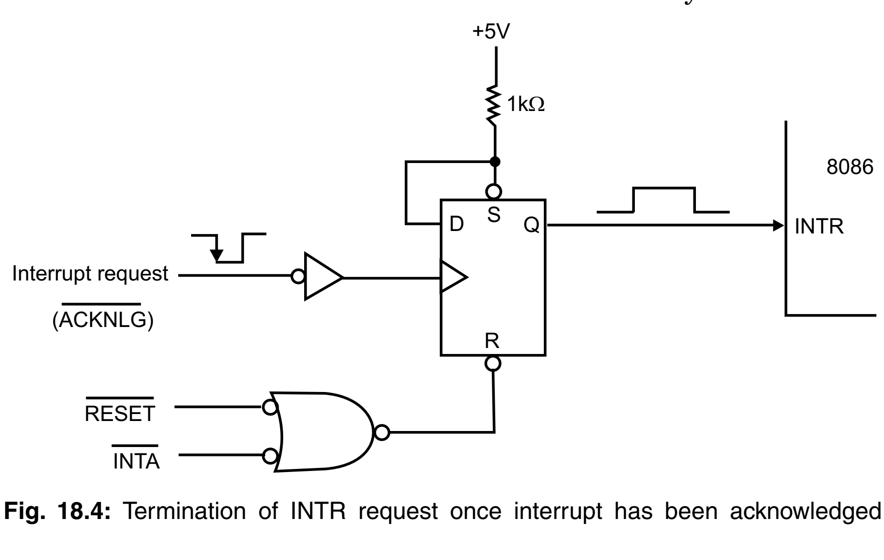


Fig. 18.4: Termination of INTR request once interrupt has been acknowledged

19. Discuss the following (a) Type 0 interrupt (b) Type 1 interrupt (c) Type 2 interrupt (d) Type 3 interrupt and (e) Type 4 interrupt.

Ans. (a) **Type 0 interrupt (or Divide-by-zero interrupt)**

If the quotient resulting from a DIV (divide) instruction or an IDIV (integer divide) instruction is too large such that it cannot be accommodated in the destination register, a divide error occurs. Then 8086 perform a Type 0 interrupt. This then passes the control to a service subroutine at addresses corresponding to IP_0 and CS_0 at 0000 H and 0002 H respectively in the pointer table.

(b) **Type 1 interrupt (Single Step interrupt)**

The single step interrupt will be enabled only if the trap flag (TF) bit is set (= 1). The TF bit can be set/reset by software.

Single step control is used for debugging in assembly language. In this mode the processor executes one instruction and then stops. The contents of various registers and memory locations can be examined. If the results are found to be ok, then a command can be inserted for execution of the next instruction. Trap flag cannot be set directly. This is done by pushing the flags on the stack, changes are made and then they are popped.

(c) **Type 2 interrupt (non-maskable NMI interrupt)**

Type 2 interrupt is the non-maskable NMI interrupt and is used for some emergency situations like power failure. When power fails, an external circuit detects this and sends an interrupt signal via NMI pin of 8086. The DC supply remains on for atleast 50 ms via capacitor banks so that the program and data remaining in RAM locations can be saved, which were being executed at the time of power failure.

(d) **Type 3 interrupt (break point interrupt)**

Type 3 interrupt is a break point interrupt. The program runs up to the break point when the interrupt occurs. This is achieved by inserting INT 3 at the point the break is desired. The ISS corresponding to Type 3 interrupt saves the register contents in the stack and can also be displayed on CRT and the control is returned to the user. This is used as a software debugging tool, like single stepping method.

(e) **Type 4 interrupt (overflow interrupt)**

The software instruction INTO (interrupt on overflow) is inserted in a program immediately after an arithmetic operation is performed. Insertion of INTO implements a Type 4 interrupt. When the signed result of an arithmetic operation on two signed numbers is too large to be stored in the destination register or else in a memory location, an overflow occurs and the OF (overflow flag) is set. This initiates INT4 instruction and the program control moves over to the starting address of the ISS, which corresponds to IP_4 and CS_4. These two are stored at address locations 0010 H and 0012 H respectively.

20. In what way the INTO instruction is different from others?

Ans. The INTO instruction is different in that no type number is needed to be mentioned.

To explain the difference, for executing any INT instruction, type no. is needed, like INT 10, INT 23, etc.

To explain further,

Opcode	Operand	Object Code	Mnemonic
INT	Type	CD 23	INT 23 H (assuming Type 23 H is employed)
INTO	none	CE	INT

21. Draw the schemes of (a) Min and (b) Max mode 8086 system external hardware interrupt interface and explain.

Ans. (a) The scheme of interconnections of Min-mode 8086 system external hardware interrupt interface is shown below in Fig. 18.5.

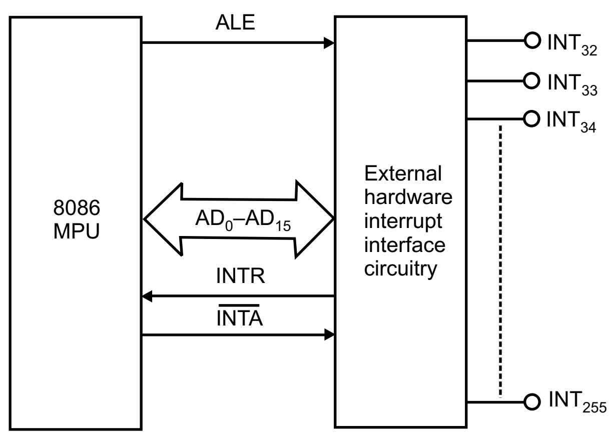


Fig. 18.5: Minimum-mode external hardware interrupt interface

The interconnecting signals to be considered for 8086 are ALE, INTR, $\overline{\text{INTA}}$ and the data bus $AD_0 - AD_{15}$.

The external device requests the service of 8086 via the INTR line. INTR is level triggered and must stay at logic 1 until recognised by the processor. Two interrupt acknowledge bus cycles are generated in response to INTR. At the end of the first bus cycle, the INTR should be removed so that it does not interrupt the 8086 a second time and the ISS can run without interruption. In this second bus cycle CPU puts the type number on the data bus of the active interrupt.

(b) The scheme of interconnections of Max-mode 8086 system external hardware interrupt interface is shown below in Fig. 18.6.

In this mode, the bus controller IC 8288 generates the $\overline{\text{INTA}}$ and ALE signals. $\overline{\text{INTA}}$ is generated when at the input of 8288, a status 000 H is applied via the status lines $\overline{S_2}\ \overline{S_1}\ \overline{S_0}$.

The $\overline{\text{LOCK}}$ signal in the figure is the bus priority lock signal and is the input to bus arbiter circuit. This circuit ensures that no other device can take control of the system buses until the presently run interrupt acknowledge cycle is completed.

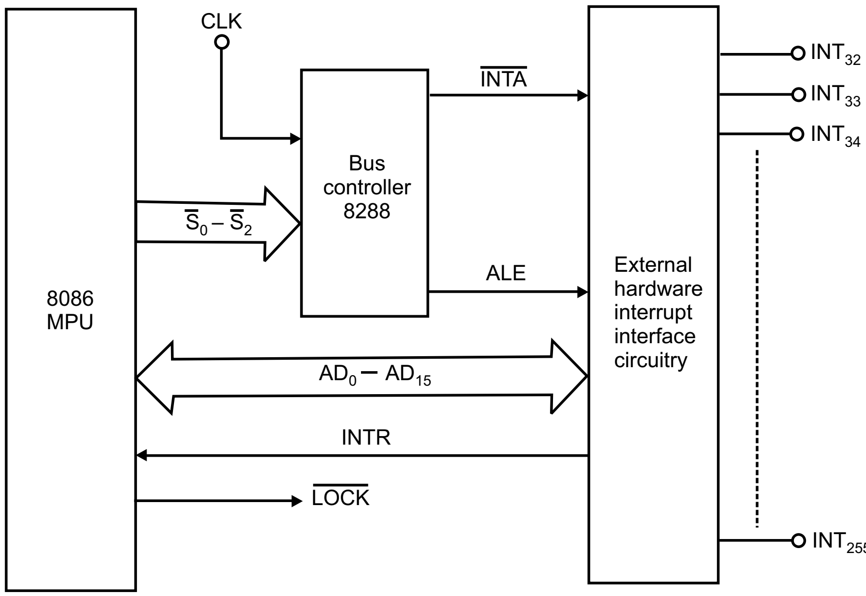


Fig.18.6: Maximum-mode 8086 system external hardware interrupt interface

19a

8288 Bus Controller

1. Draw the pin diagram of 8288.

Ans. The diagram of 8288 is shown in Fig. 19a.1.

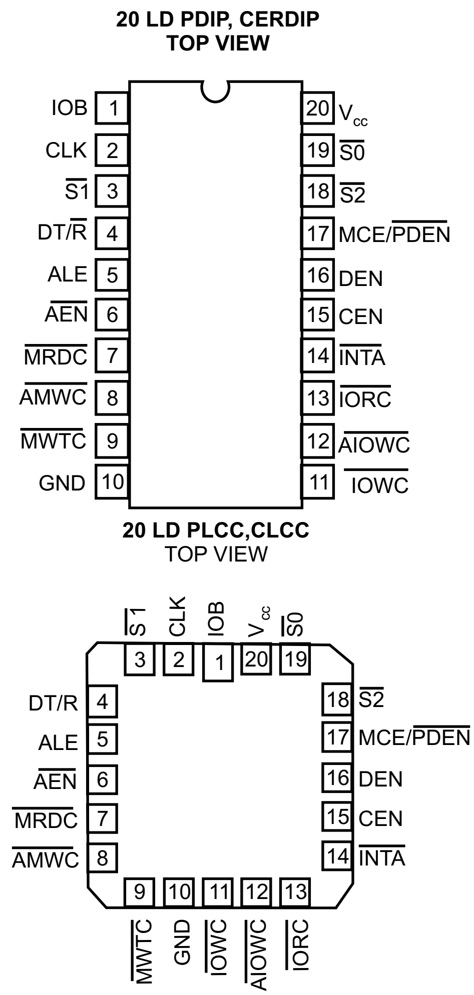


Fig. 19a.1: 8288 pin diagram

2. Draw the functional block diagram of 8288.

Ans. The functional block diagram of 8288 is shown in Fig. 19a.2.

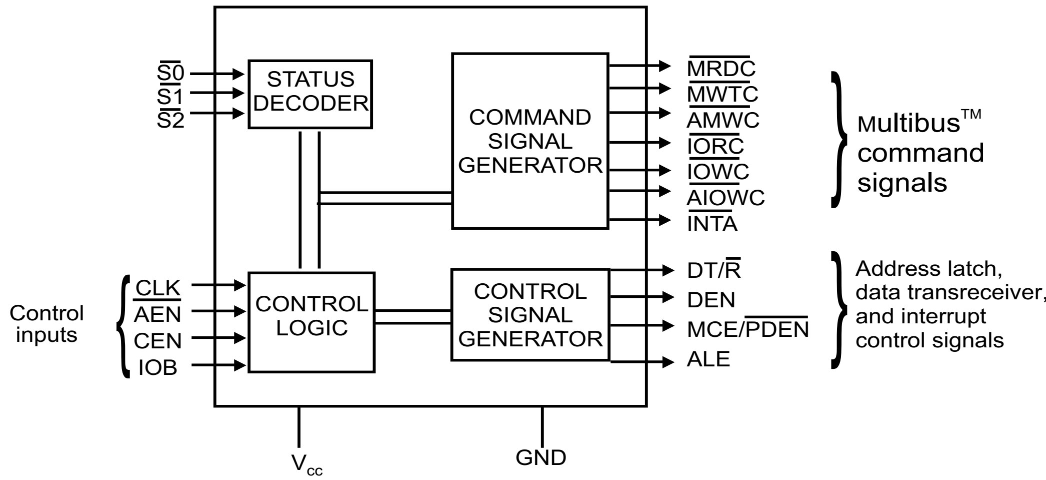


Fig. 19a.2: 8288 block diagram

3. Is 8288 always used with 8086?

Ans. No, the bus controller IC 8288 is used with 8086 when the latter is used in MAX mode.

4. What are the inputs to 8288?

Ans. There are two sets of inputs—the first set is the status inputs $\overline{S_0}$, $\overline{S_1}$ and $\overline{S_2}$. The second set is the control inputs having the following signals: CLK, $\overline{\text{AEN}}$, CEN and IOB.

5. What are the output signals from 8288?

Ans. There are two sets of output signals—Multibus command signals and the second set includes the bus control signals—Address Latch, Data Transreceiver and Interrupt Control Signals.

The multibus command Signals are the Conventional $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$ and $\overline{\text{IOW}}$ signals which have been renamed as $\overline{\text{MRDC}}$, $\overline{\text{MWTC}}$, $\overline{\text{IORC}}$, $\overline{\text{IOWC}}$, where the suffix 'C' stands for command. $\overline{\text{INTA}}$ signal is also included in this.

Two more signals—$\overline{\text{AMWC}}$ and $\overline{\text{AIOWC}}$ are the advanced memory and I/O write commands. These two output signals are enabled one clock cycle earlier than normal write commands. Some memory and I/O devices require this wider pulse width.

The bus control signals are DT/$\overline{\text{R}}$, DEN, ALE and MCE/$\overline{\text{PDEN}}$. The first three are identical to 8086 output signals when operated in the MIN mode—with the only difference here is that the DEN output signal of 8288 is an active high signal.

MCE/$\overline{\text{PDEN}}$ (Master Cascade Control/Peripheral Data Enable) is an output signal having two functions—I/O bus control or system bus control. When this signal status is low, its function is identical to DEN signal and it operates in I/O mode. When high, this signal ensures the sharing of the system buses by other processors connected to the system.

In the system bus control mode, the signals $\overline{AEN}$ (address enable) and IOB both have to be low. This then permits more than one 8288 and 8086 to be interfaced to the same set of system buses. In this case, the bus arbiter IC 8289 selects the active processor by enabling only one 8288, via the $\overline{AEN}$ input. In this system bus mode MCE/$\overline{PDEN}$ signal becomes MCE-Master Cascade Control and is used during an interrupt sequence to read the address from a master priority interrupt controller (PIC).

The operating modes of 8288 are determined by CEN (command enable), IOB, (I/O bus) and $\overline{AEN}$ signals and shown in Table 19a.1.

Table 19a.1: 8288 I/O and System bus modes

CEN	IOB	$\overline{AEN}$	Description
1	1	×	I/O bus mode, all control signals enable; MCE/$\overline{PDEN}$ = $\overline{PDEN}$
1	0	1	System bus mode, all control signals disabled. The bus is busy—i.e., controlled by another bus master.
1	0	0	System bus mode. All control signals active, the bus is free to use; MCE/$\overline{PDEN}$ = MCE
0	×	×	All command signals and the DEN and $\overline{PDEN}$ outputs are disabled (Open-circuited)

6. Discuss the status pins $\overline{S_2}$, $\overline{S_1}$ and $\overline{S_0}$.

Ans. These are three input pins for 8288 and come from the corresponding pins of 8086 (its output pins). The command-decode definitions for various combinations of the three signals are shown in Table 19a.2.

Table 19a.2: Command Decode Definition

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Processor state	82C88 command
0	0	0	Interrupt Acknowledge	$\overline{INTA}$
0	0	1	Read I/O Port	$\overline{IORC}$
0	1	0	Write I/O Port	$\overline{IOWC}$, $\overline{AIOWC}$
0	1	1	Halt	None
1	0	0	Code Access	$\overline{MRDC}$
1	0	1	Read Memory	$\overline{MRDC}$
1	1	0	Write Memory	$\overline{MWTC}$, $\overline{AMWC}$
1	1	1	Passive	None

7. Discuss the three pins (a) IOB (b) CEN and (c) $\overline{\text{AEN}}$ of 8288.

Ans. (a) IOB stands for input/output bus mode and is an input signal for 8288. When IOB = high, 8288 functions in the I/O bus mode and when IOB = low, 8288 functions in the system bus mode.

(b) CEN stands for command enable and is an input signal for 8288. When CEN = low, all command outputs of 8288 and the DEN and the PDEN control outputs are forced into active state and not tri-stated. This feature is utilised for memory partitioning implementation. This also eliminates address conflicts between system bus devices and resident bus devices. Again, when CEN = high, these outputs are in the enabled state.

(c) $\overline{\text{AEN}}$ stands for address enable and is an input signal for 8288. This signal enables command outputs of 8288 a minimum of 110 ns (and a maximum of 250 ns) after it becomes low (i.e., active). If $\overline{\text{AEN}}$ = 1, then command output drivers are put to tri-state. In the I/O bus mode (IOB = 1) $\overline{\text{AEN}}$ signal does not affect the command lines.

19b

8087 Numeric Data Processor

1. Draw the pin connection diagram of 8087.

Ans. The pin diagram of 8087 is shown in Fig. 19b.1.

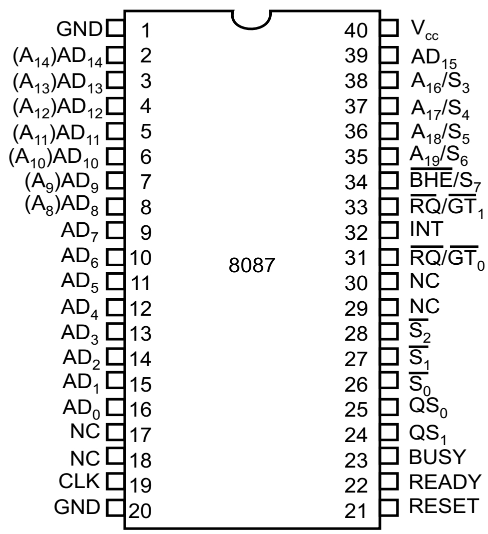


Fig. 19b.1: 8087 Pin Diagram

2. What are the characteristics of 8087 NDP?

Ans. The following are the characteristic features of 8087 NDP:

- It can add arithmetic, trigonometric, exponential and logarithmic instructions to the 8086 instruction set for all data types.
- 8087 can handle seven data types. These are : 16, 32, 64-bit integers, 32, 64, 80-bit floating point and 18–digit BCD operands.
- It has three clock speeds: 5 MHz (8087), 8 MHz (8087–2) and 10 MHz (8087–1)
- It can add 8 × 80-bit individually addressable register stack to the 8086 architecture.
- Multibus system compatiable interface.
- Seven numbers built-in exception handling functions.
- Compatible with IEEE floating point standard 754.
- It adds 68 mnemonics to the 8086 instruction set.
- Fabricated with HMOS III technology and packaged in a 40-pin cerdip package.

3. Draw the architecture of 8087.

Ans. The architecture of 8087 is shown below in Fig. 19b.2.

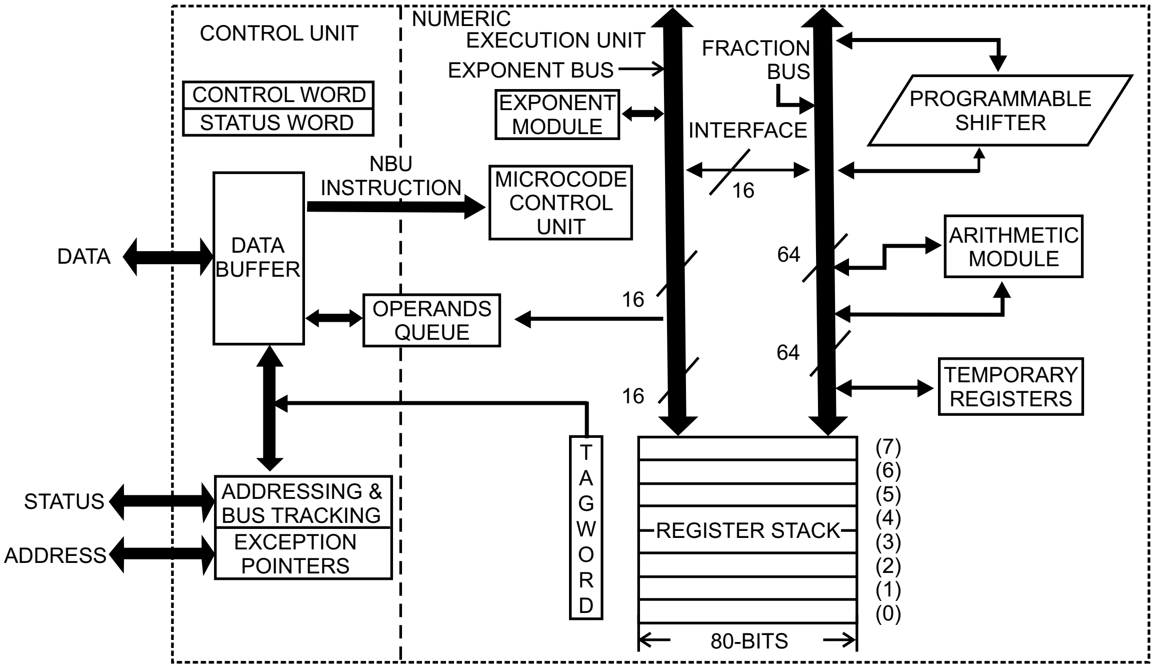


Fig. 19b.2: 8087 Internal block diagram (Intel Corporation)

4. How does 8086 view 8087 NDP?

Ans. At the hardware level, 8086 treats 8087 as an extension to its own CPU capability—providing for registers, data types, control and instruction capabilities.

At the software level, both 8086 and 8087 are viewed as a single unified processor.

5. Discuss A_{16}/S_3 to A_{19}/S_6 signals.

Ans. When 8086 CPU is in control of buses, these four lines act as input lines, which 8087 monitors.

When 8087 controls the buses, these are the four most significant address lines (A_{19} to A_{16}) for memory operations during T_1. Also, during T_2, T_3, T_w and T_4, status information is available on these four lines. During these states, S_6, S_4 and S_3 are high while S_5 is always low.

6. Discuss the $\overline{S2}$, $\overline{S1}$ and $\overline{S0}$ signals.

Ans. These three signals act as both input or output pins.

When the CPU is active, i.e., CPU is in control of buses, these three signals act as input pins to 8087. 8087 monitors these signals coming from 8086.

When 8087 is driving, these three status lines act as output pins and are encoded as follows:

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	
0	×	×	Unused
1	0	0	Unused
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive

The status lines are driven active during T_4, remains valid during T_1 and T_2 and returns to passive state (111) in T_3 or T_w, when READY is high. The status on these lines are monitored by 8288 to generate all memory access control signals.

7. What language 8087 supports?

Ans. 8087 supports the high level languages of 8086 which are:
ASM-86, PL/M-86, FORTRAN-86 and PASCAL-86.

8. How 8-bit and 16-bit hosts are taken care of by 8087?

Ans. For 8-bit and 16-bit hosts, all memory operations are performed as byte or word operations respectively.

8087 determines the bus width during a system reset by monitoring pin 34 ($\overline{BHE}/S7$) of 8087. For 16-bit hosts a word access from memory locations FFFF0 H (a dedicated location) is performed and pin 34 ($\overline{BHE}$) will be low while, for a 8-bit host, a byte access from the same memory location FFFF0 H is performed and pin 34 ($\overline{SS0}$) will be high.

9. What is the operating frequency of 8087?

Ans. 8087 operates at frequencies of 5 MHz, 8 MHz and 10 MHz.

10. What is the utility of having NDP 8087 along with 8086?

Ans. 8086 is a general purpose microprocessor suitable mostly for data processing applications. But in cases where scientific and other calculation-intensive applications are involved, 8086 fails with its integer arithmetic and four basic math function capabilities.

8087 can process fractional number system and transcendental math functions with its special coprocessor instructions in parallel with the host CPU—thus relieving 8086 with these tasks. In Intel number scheme, 8086/8087 system is known as an iAPX86/20 two-chip CPU.

11. What are the two units in NDP and what do they do?

Ans. There are two units in 8087—a Control Unit (CU) and a Numeric Execution Unit (NEU).

These two units can operate independently i.e., asynchronously—like the BIU and EU of 8086.

The CU receives, decodes instructions, reads and writes memory operands and executes all other control instructions. NEU does the job of arithmetic processing. The control unit establishes the communication between the CPU and memory and coordinates the internal processor activities.

12. How 8086 communicates with NDP 8087?

Ans. The opcodes for 8087 are written into memory along with those for 8086. As 8086 (host) fetches instructions from memory, 8087 also does the same. These prefetched instructions are put into the queue of both 8086 and 8087, while $\overline{S_0}$, $\overline{S_1}$ and $\overline{S_2}$ provide the information on bus status.

When an ESC instruction is encountered by the host (8086), it calculates the effective address (EA) and performs a 'dummy' read cycle. The data read is not stored. However, a read or write cycle is performed by 8087 from this EA. 8087 does not have any facility to generate EA on its own. If the coprocessor instruction does not need any memory operand, then it is directly executed by 8087.

Several data formats as used by 8087 require multiple word memory operands. In such cases, 8087 need to have the buses under its own control. This is achieved via RQ/GT line. The sequence is as follows:

- 8087 sends out a low going pulse on its RQ/GT pin (connected to the RQ/GT pin of 8086) of one clock pulse duration.
- 8087 waits for the grant pulse from the host (8086).
- When it is received by 8087, it increments the address and outputs the incremented address on the address bus.
- 8087 continues memory-read or memory-write signals until all the instructions meant for 8087 are complete.
- At this point, another low going pulse is sent out by 8087 (to 8086) on its RQ/GT line, to let 8086 know that it can have the buses back again.

When the NEU (numeric execution unit) begins executing arithmetic, logical, transcendental and data transfer instructions, it (8087) pulls up BUSY signal. This output signal of 8087 is connected to the $\overline{\text{TEST}}$ input of 8086. This forces 8086 to wait until the $\overline{\text{TEST}}$ input of 8086 goes low (i.e., BUSY output of 8087 becomes low).

The microcode control unit of 8087 generates the control signals for execution of instructions while the 'programmable shifter' shifts the operands, during the execution of instructions like FMUL and EDIV. The data bus interfaces the internal data bus of 8087 with the data bus of the host (8086).

13. What INT output signal does?

Ans. This interrupt output is utilised by 8087 to indicate that an unmasked exception has been received during excitation by 8087. This is normally handled by 8259A.

14. Discuss the $\overline{\text{BHE}}$/S7 signal of 8087.

Ans. During T_1 cycle, $\overline{\text{BHE}}/S7$ output pin used to enable data on the higher byte of the 8086 data bus. During T_2, T_3, T_w and T_4, this pin becomes the status line S_7.

15. What kind of errors/exception conditions 8087 can check?

Ans. During execution of instructions, 8087 can check the following kind of errors/exception conditions:

- **Invalid operation:** This includes the attempt to calculate the square root of a negative number or say to take out an operand from an empty register. Also included in this category are stack overflow, stack underflow, indeterminate form or the use of a non-number as an operand.

- **Overflow:** Exponent of the result is too large for the destination real format.
- **Zero divide:** Arises when divisor is zero while the dividend is a non-infinite, non-zero number.
- **Denormalised:** It arises when an attempt is made to operate on an operand that is yet to be normalised.
- **Under flow:** Exponent of the result is too small to be represented.
- **Precision:** In case the operand is not made to represent in the destination format, causing 8087 to round the result.

16. What does 8087 do in case an exception occurs?

Ans. 8087 sets the appropriate flag bit in the status word in case of occurrence of any one of the exception conditions. The exception mask in the control register is then checked and if the mask bit is set i.e., masked, then a built-in fix-up procedure is followed.

If the exception is unmasked (i.e., mask bit = 0), then user-written exception handlers take care of such situations. This is done by using the INT pin which is normally connected to one of the interrupt input pins of 8259A PIC.

17. Describe the register set of 8087.

Ans. The register set of 8087 comprises the following:

- Eight data registers—each 80-bits wide
- A tag field R_1–R_8.
- Five control/status registers.

The register set of 8087 is shown in Fig. 19b.3.

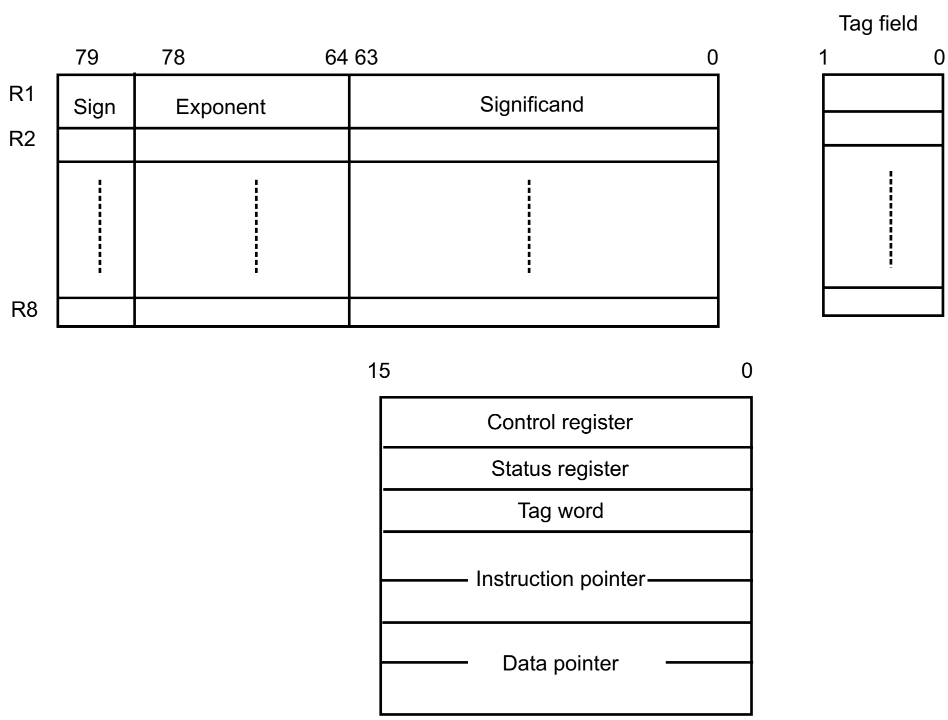


Fig. 19b.3: The register set of 8087

The eight data registers, residing in NEU, can be used as a stack or a set of general registers. These registers are divided into three fields—sign (1-bit), exponent (15-bits) and significand (64-bits). Corresponding to each of these eight registers, there is a two bit TAG field to indicate the status of contents.

The control and status registers are shown in Fig. 19b.4. The top bits in the status register indicate the register currently at the top of the register stack. Bits 0–5 indicate the 'exception' status, while the condition code bits C2 – C0 are set by various 8087 operations similar to the flags within the CPU.

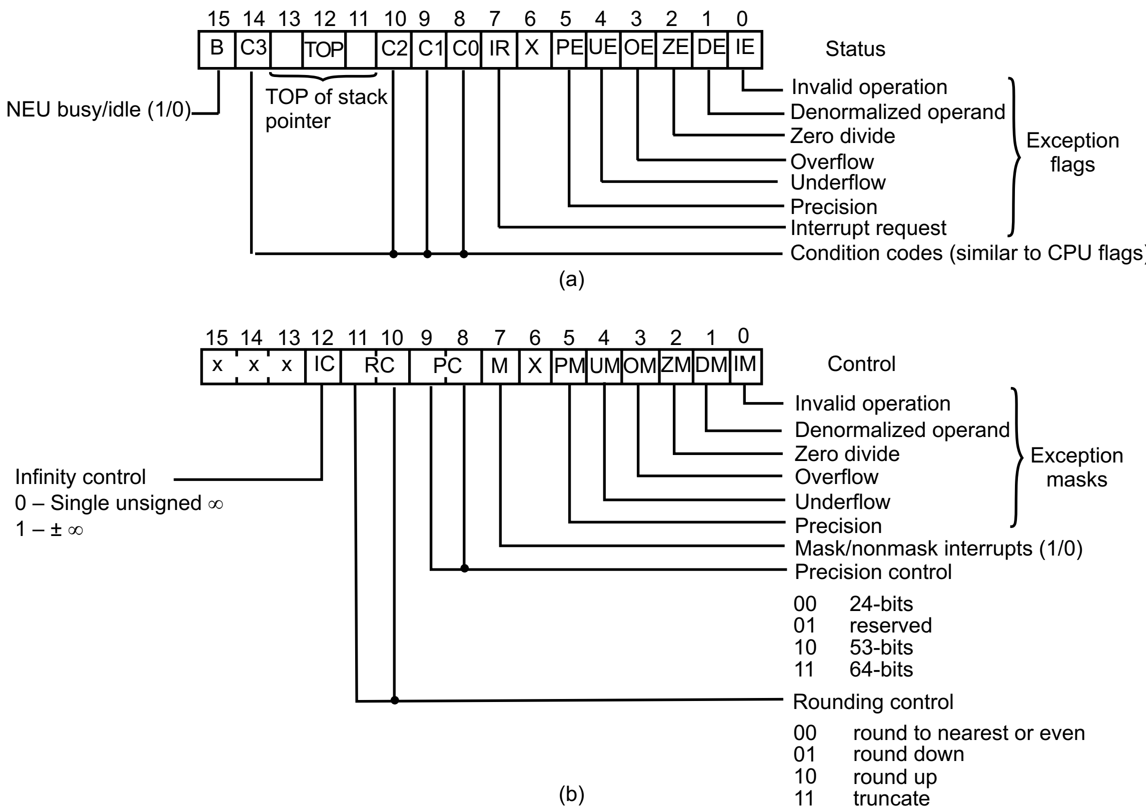


Fig. 19b.4: (a) The 8087 status word can be read to check the state of the NDP
(b) The control word can be written to select various processing options

Bits 0 – 5 of the control word register allow any of the exception cases to be masked. Bit 6 is a 'don't care' bit while bit 7 must be reset (low) for the interrupts to be accepted. The upper bits of the control register defines type of infinity, rounding and precision to be used when 8087 performs the calculations.

The control register defines the type of infinity, rounding and precision to be used when NDP performs and executes the instructions. For the interrupts to be accepted, bit 7 must be reset while bits 0–5 allow any of the exception cased to be masked.

Both the Data & Instruction pointer registers are 20-bits wide. Whenever the NEU executes an instruction, CU saves the opcode and memory address corresponding to this instructions and also its operand in these registers. For this, a control instruction is needed for their contents to be stored in memory and is utilised for program debugging.

Initially, Data registers of 8087 are considered to be empty, unlike the CPU registers. The data written into these registers can be valid (i.e., the register holds a valid data in temporary real format), zero or special (indefinite due to error). Each data register has a tag field associated with the corresponding register. Each tag field is two bit wide and contains one of the four states that each of the data registers can hold. The eight numbers of 2-bit tag field again are grouped into a single tag word—it is thus 16-bit wide. It optimises the NDP's performance under certain conditions and normally needed by the programmer.

19c

8089 I/O Processor

1. Draw the pin connection diagram of 8089.

Ans. The pin connection diagram of 8089 is shown in Fig. 19c.1.

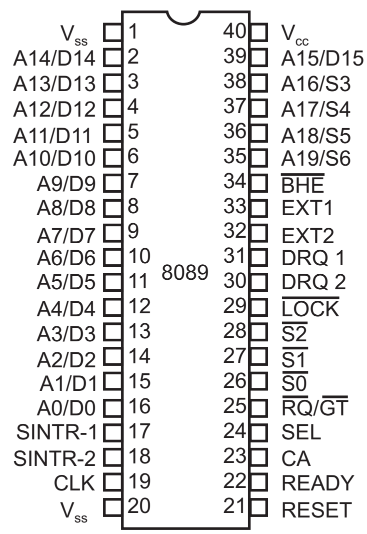


Fig.19c.1: 8089 pin diagram

2. Draw the functional block diagram of 8089.

Ans. The functional block diagram of 8089 is shown in Fig. 19c.2.

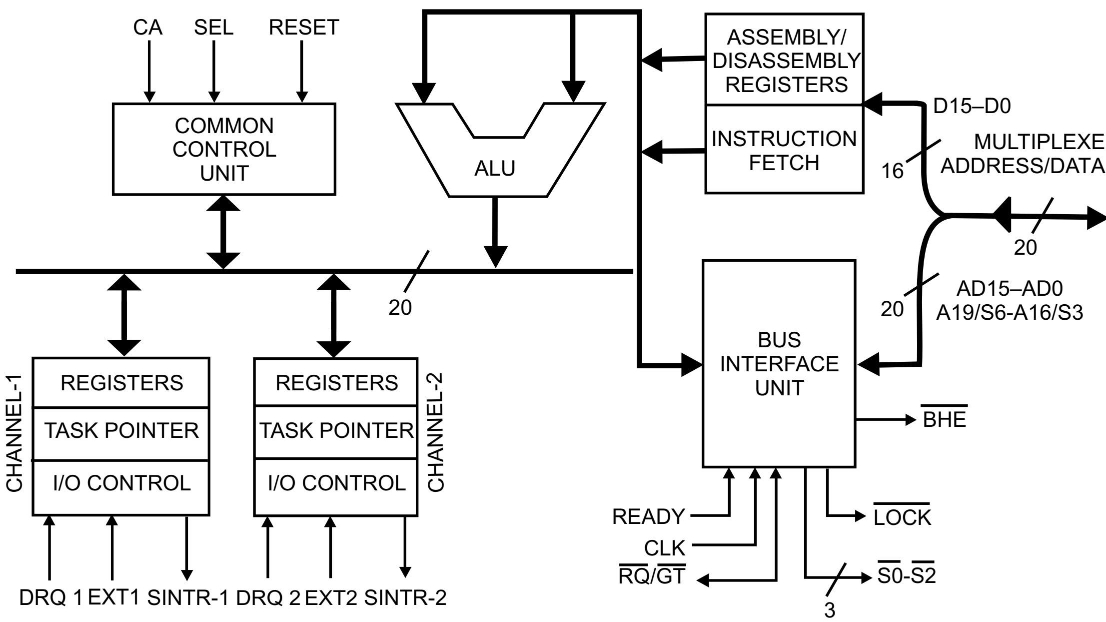


Fig. 19c.2: 8089 block diagram

3. Write down the characteristic features of 8089.

Ans. The characteristic features of 8089 are as follows:

- Very high speed DMA capability—I/O to memory, memory to I/O, memory to memory and I/O to I/O.
- 1 MB address capability.
- iAPX 86, 88 compatible.
- Supports local mode and remote mode I/O processing.
- Allows mixed interface of 8-and 16-bit peripherals, to 8-and 16-bit processor buses.
- Multibus compatible system interface.
- Memory based communications with CPU.
- Flexible, intelligent DMA functions, including translation, search, word assembly/ disassembly.
- Supports two I/O channels.

4. Indicate the data transfer rate of 8089 IOP.

Ans. On each of the two channels of 8089, data can be transferred at a maximum rate of 1.25 MB/second for 5MHz clock frequency.

5. Mention a few application areas of 8089.

Ans. A few of the application areas of 8089 are:

- File and buffer management in hard disk/floppy disk control.
- Provides for soft error recovery routines and scan control.
- CRT control such as cursor control and auto scrolling made simple with 8089.
- Keyboard control, communication control, etc.

6. Compare 8089 IOP with 8255 PPI and 8251 USART.

Ans. 8089 IOP is a front-end processor for the 8086/88 and 80186/88. In a way, 8089 is a microprocessor designed specifically for I/O operations. 8089 is capable of concurrent

operation with the host CPU when 8089 executes a program task from its own private memory.

8255 PPI and 8251 USART are peripheral controller chips designed to simplify I/O hardware design by incorporating all the logic for parallel (in case of 8255) or serial (in case of 8251) ports in one single package. These two chips need to be initialized for them to be used. But data transfer is controlled by CPU.

8089 IOP does not have any built-in I/O ports, not it is a replacement for 8255 or 8251. For I/O operations, 8089 executes the I/O related softwares, previously run by CPU (when no 8089 was used).

Thus in situations where I/O related operations are in a majority, 8089 does all these jobs independent of CPU. Once done, the host CPU communicates with 8089 for high speed data transfer either way.

7. Does 8089 generate any control signals.

Ans. No, 8089 does not output control bus signals: $\overline{IOW}$, $\overline{IOR}$, $\overline{MEMR}$, $\overline{MEMW}$, DT/$\overline{R}$, ALE and DEN. These signals are encoded into $\overline{S0}-\overline{S2}$ signals, which are output pins for 8089 and are connected to the corresponding pins of 8288 bus controller and 8289 bus arbiter to generate memory and I/O control signals. The bus controller then outputs all the above stated control bus signals. The $\overline{S0}-\overline{S2}$ signals are encoded as follows:

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	
0	0	0	Instruction fetch; I/O space
0	0	1	Data fetch; I/O space
0	1	0	Data store; I/O space
0	1	1	Not used
1	0	0	Instruction fetch; system memory
1	0	1	Data fetch; system memory
1	1	0	Data store; system memory
1	1	1	Passive

These signals change during T4 if a new cycle is to be entered. The return to passive state in T3 or TW indicates the end of a cycle. These pins float after a system reset—when the bus is not required.

8. Explain the utility of $\overline{\textbf{LOCK}}$ signal.

Ans. It is an output signal and is set via the channel control register and during the TSL instruction. This pin floats after a system reset—when the bus is not required.

The $\overline{LOCK}$ signal is meant for the 8289 bus arbiter and when active, this output pin prevents other processors from accessing the system buses. This is done to ensure that the system memory is not allowed to change until the locked instructions are executed.

9. Explain DRQ1-2 and EXT1-2 pins.

Ans. DRQ and EXT stand for Data Request and External Terminate, both being input pins—DRQ1 and EXT1 for channel 1 and DRQ2 and EXT2 for channel 2.

DRQ is used to initiate DMA transfer while EXT for termination of the same. A high on DRQ1 tells 8089 that a peripheral is ready to receive/transfer data via channel 1. DRQ must be held active (= 1) until the appropriate fetch/stroke is initiated.

A high on EXT causes termination of current DMA operation if the channel is so programmed by the channel control register. This signal must be held active (= 1) until termination is complete.

10. Explain the common control unit (CCU) block.

Ans. 8089 IOP has two channels. The activities of these two channels are controlled by CCU. CCU determines which channel—1 or 2 will execute the next cycle. In a particular case where both the channels have equal priority, an interleave procedure is adopted in which each alternate cycle is assigned to channels 1 and 2.

11. Explain the purpose of assembly/disassembly registers.

Ans. This permits 8089 to deal with 8-or 16-bit data width devices or a mix of both. In a particular case of an 8–bit width I/O device inputting data to a 16-bit memory interface, 8089 capture two bytes from the device and then write it into the assigned memory locations—all with the help of assembly/disassembly register.

12. Explain SINTR pin.

Ans. SINTR stands for signal interrupt. It is an output pin from 8089 and there are two such output pin SINTR1 and SINTR2—for channel 1 and 2 respectively.

Like 8087, 8086 does not communicate with 8089 directly. Normally, this takes place via a series of commonly accessible message blocks in system memory.

SINTR pin is another method of such communication. This output pin of 8087 can be connected directly to the host CPU (8086) or through an 8259 interrupt controller. A high on this pin alerts the CPU that either the task program has been completed or else an error condition has occurred.

13. Elaborate on the communication between CPU and IOP with the help of communication data structure hierarchy.

Ans. Communication between the CPU and IOP takes place through messages in shared memory and consists of five linked memory message blocks (ABCDE or $ABCD'E'$) with ABC representing the initialisation process. The process of initialisation begins with 8089 IOP receiving a reset at its RESET input. The following occurs in sequence:

- On the falling edge of CA, the SEL input is sensed. SEL = 0/1 represent Master (Remote)/Slave (Local) configuration. (To note: during any other CA, the SEL line indicates selection of CH1/CH2 depending on SEL = 0/1 respectively).
- 5 bytes of information from system memory starting from FFFF6 H is read into 8089. The first byte determines the width of the *system bus*. The subsequent bytes are then read to get the system configuration pointer (SCP) which gives the locations of the system configuration block (SCB).
- The first byte existing at the base of SCB is read off, which determines the width of the 8089's *Private Bus* and the operating mode of $\overline{RQ}/\overline{GT}$ is defined. The base (or starting) address of control block (CB) is then read.
- The BUSY flag in CB is removed, signalling the end of the initialisation process.

It should be noted that the address of SCP—the system configuration pointer resides in ROM and is the only one to have fixed address in the hierarchy. Since SCB resides in RAM, hence it can be changed to accommodate additional IOPs, to be inducted into the system.

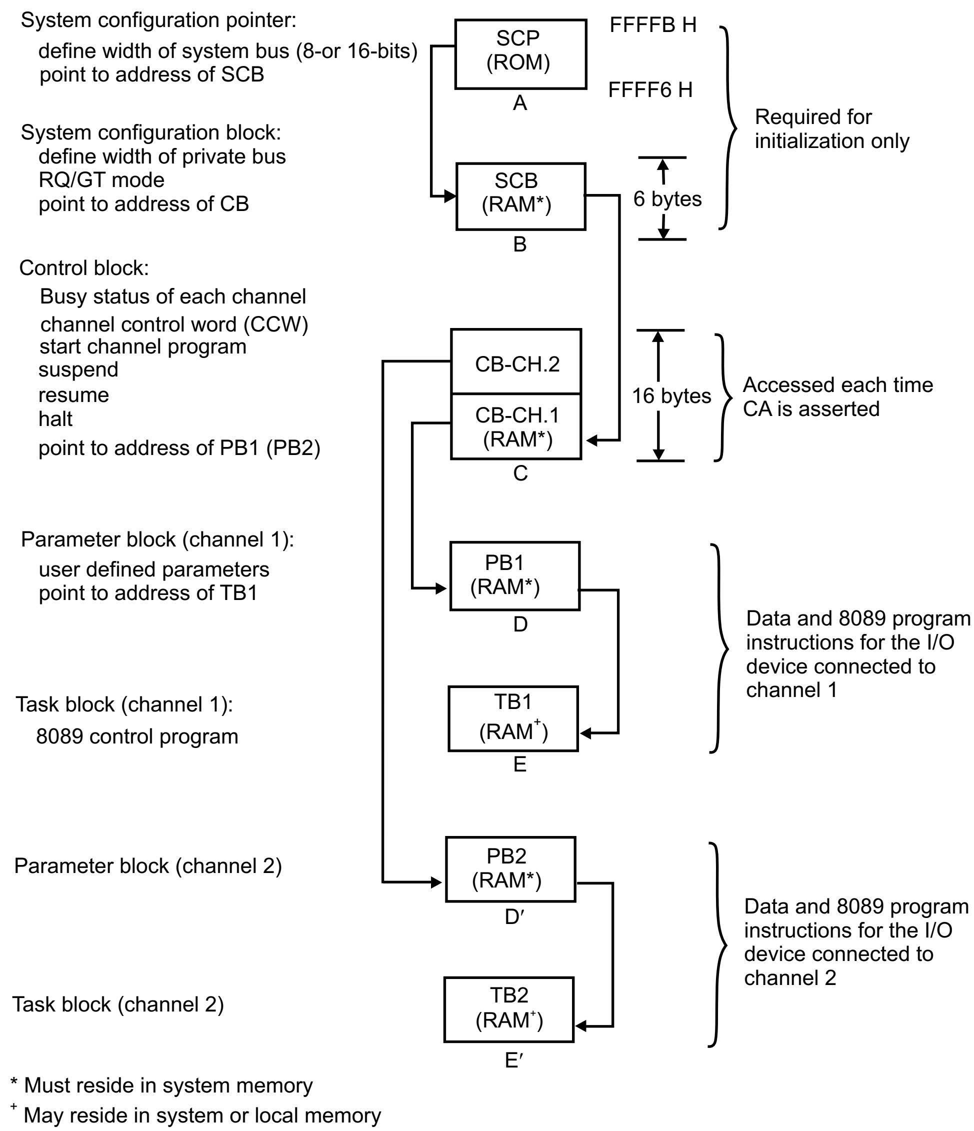


Fig.19c.3: The 8089 uses a series of five linked memory message blocks.
All except the task block must be located in memory accessible to the 8089 and the host processor.

Once initialisation is over, any subsequent hardware CA input to IOP accesses the control block (CB) bytes for a particular channel—the channel (1 or 2) which gets selected depends on the SEL status. First the CCW (Channel Control Word) is read. Next the base address for the parameter block (PB) is read. This is also called data memory. Except the first two words, this PB block is user defined and is used to pass appropriate parameters to IOP for task block (TB), also called program memory. The task block can be terminated/restarted by the CPU.

This hierarchical data structure between the CPU and IOP gives modularity to system design and also future compatibility to future end users.

14. Show the channel register set model and discuss.

Ans. The channel register set for 8089 IOP is shown in Fig.19c.4.

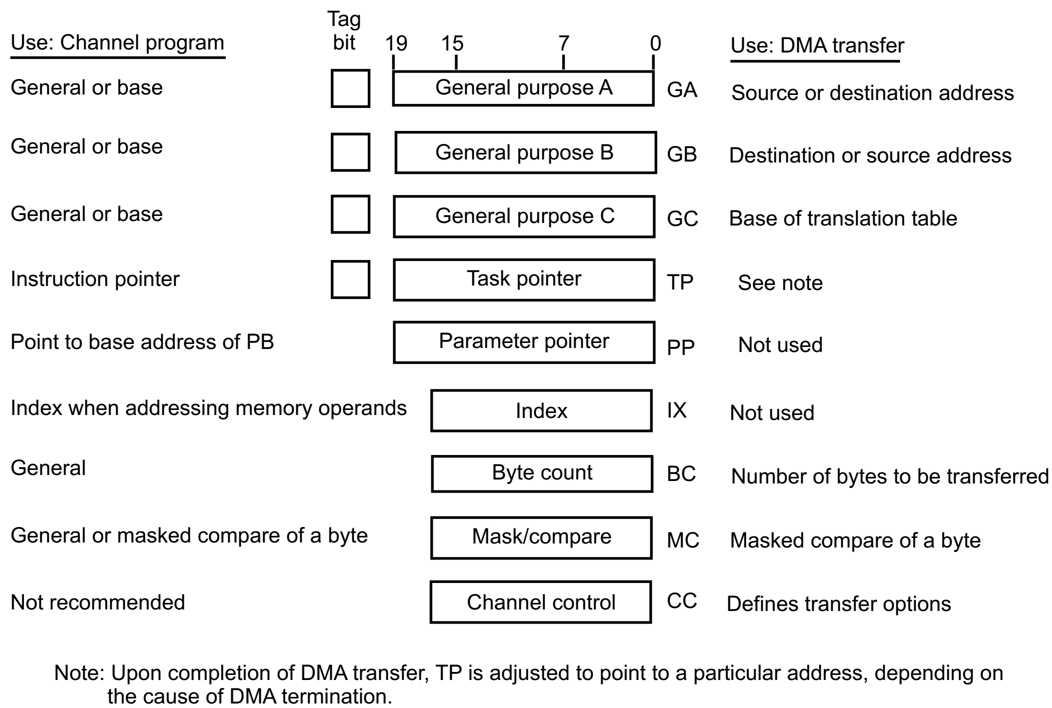


Fig. 19c.4: The channel register set of 8089

Registers GA, GB and GC and TP may address 1 MB of memory space (with tag bit = 0) or 64 KB of I/O space (with tag bit = 1). These four registers as also PP are called pointer registers.

The rest four registers—IX, BC, MC and CC are all 16 bits wide. Several DMA options can be programmed with the help of register CC.

15. Mention the addressing modes of 8089 IOP.

Ans. 8089 IOP has six different addressing modes. These are:

- Register addressing
- Immediate addressing
- Offset addressing
- Based addressing
- Indexed addressing and
- Indexed with auto increment addressing.

19d

8289 Bus Arbiter

1. Draw the pin connection diagram of 8289.

Ans. The following is the connection diagram of 8289.

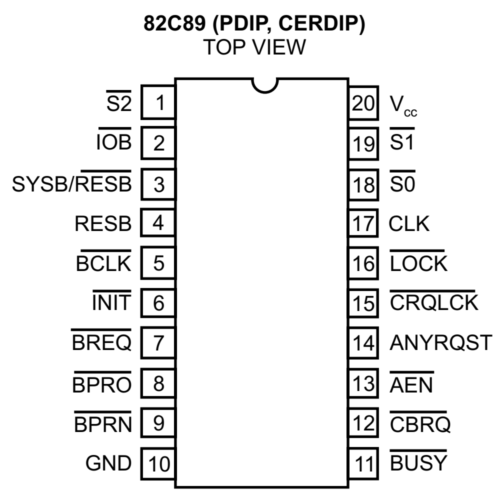


Fig. 19d.1: 8289 pin diagram

2. Draw the block diagram of 8289.

Ans. Figure 19d.2 shows the block diagram of 8289.

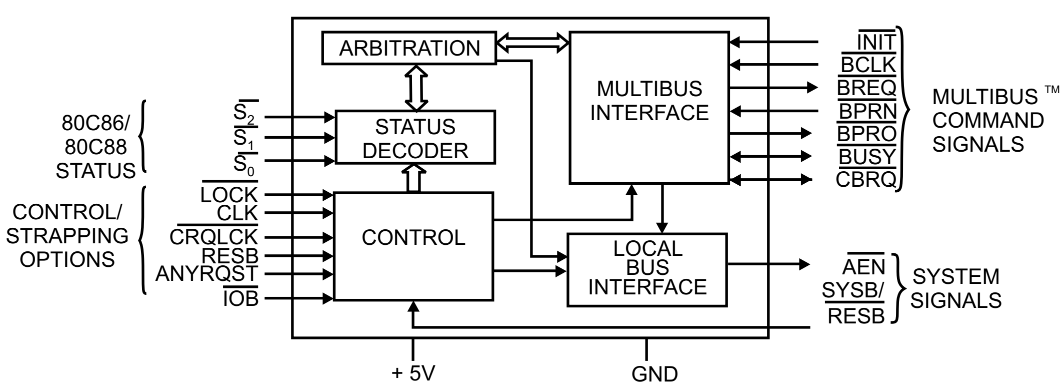


Fig. 19d.2: 8289 block diagram

3. Explain how 8289 bus arbiter operates in a multi-master system.

Ans. In MAX mode 8086 processor is interfaced with 8289 bus arbiter, along with bus controller IC 8288 in a multi-master system bus configuration. When the processor does not use the system buses, bus arbiter forces the bus driver output in the high impedance state. The bus arbiter allows the bus controller, the data transreceivers and the address latches to access the system bus.

On a multi-master system bus, the bus arbiter is responsible for avoiding the bus contention between bus masters.

4. How the arbitration between bus masters works?

Ans. The bus is transferred to a higher priority master when the lower priority master completes its task. Lower priority masters get the bus when a higher priority one does not seek to access the bus, although with the help of ANYRQST input, the bus arbiter will allow to surrender the bus to a lower priority master from a higher one. The bus arbiter maintains the bus and is forced off the bus only under HALT instruction.

5. Discuss $\overline{\text{LOCK}}$ and $\overline{\text{CRQLCK}}$ signals.

Ans. Both are active low input signals, the second one standing for Common Request Lock.

A *processor* generated active low signal on the $\overline{\text{LOCK}}$ output pin is connected to $\overline{\text{LOCK}}$ input pin of 8289, and prevents the arbiter from surrendering the multi-master system bus to any other bus arbiter, regardless of its priority.

$\overline{\text{CRQLOCK}}$: An active low on this input pin prevents the arbiter from surrendering the multi-master system bus to any other bus arbiter IC after being requested through $\overline{\text{CBRQ}}$ input pin.

6. Discuss $\overline{\text{AEN}}$ and $\overline{\text{INIT}}$ pins of 8289.

Ans. Both are active low signals, with the former being an output signal and the latter an input signal.

A high on $\overline{\text{AEN}}$ signal puts the output drivers of 8288 bus controller, address latches and the 8284 clock generator into high impedance state.

An active signal (= 0) on $\overline{\text{INIT}}$ input resets all bus arbiters on the multi-master system bus. After initialisation is over, no arbiter can use the said bus.

7. Discuss RESB and SYSB/$\overline{\text{RESB}}$ pins of 8289.

Ans. Both are input pins for 8289 bus arbiter. RESB and SYSB stand for Resident Bus and System Bus respectively. When RESB is high, the multi-master system bus is requested or surrendered which is a function of SYSB/$\overline{\text{RESB}}$ input. When RESB is put to low, SYSB/$\overline{\text{RESB}}$ input is ignored.

Again, the arbiter requests the multi-master system bus in the System/Resident Mode when SYSB/$\overline{\text{RESB}}$ is high and allows the bus to be surrendered when this pin is low.

8. Explain $\overline{\text{BREQ}}$ and $\overline{\text{BPRO}}$ pins.

Ans. Both are active low output pins. The first one stands for Bus Request while the second one stands for Bus Priority Out.

$\overline{\text{BREQ}}$ is used in the parallel priority resolving scheme which a particular arbiter activates to request the use of muti-master system bus.

$\overline{\text{BPRO}}$ is used in the serial priority resolving scheme and it is daisy-chained to $\overline{\text{BPRN}}$ of the just next lower priority arbiter.

9. Explain $\overline{\text{BPRN}}$ pin.

Ans. It is an active low input and stands for Bus Priority In. When a low is returned to the arbiter, it instructs the same that it may acquire the multi-master system bus on the falling edge of $\overline{\text{BCLK}}$. The active condition of $\overline{\text{BPRN}}$ indicates that it is the highest priority arbiter presently on the bus. If an arbiter loses its $\overline{\text{BPRN}}$ active signal, it means that it has lost its bus priority to a higher priority arbiter.

10. Explain $\overline{\text{BUSY}}$ pin.

Ans. It is an active low input-output pin. With the availability of multi-master system bus, the highest priority arbiter seizes the bus, as determined by the status of $\overline{\text{BPRN}}$ input. This thus keeps the other arbiters off the bus. When the particular arbiter has completed its job, it releases the $\overline{\text{BUSY}}$ signal, thereby allowing the next highest arbiter to seize the bus.

11. Explain ANYRQST and $\overline{\text{CBRQ}}$ pins.

Ans. ANYRQST stands for Any Request—it is an active high input pin. $\overline{\text{CBRQ}}$ is, on the other hand, an input/output active low pin and stands for Common Bus Request.

An active signal on ANYRQST would enable the multi-master system bus to be handed over to an arbiter—even if it has lower priority.

When acting as an input, an active condition on $\overline{\text{CBRQ}}$ tells the arbiter of the presence of other lower priority arbiters in the multi-master system bus.

The $\overline{\text{CBRQ}}$ pins of the particular arbiters which would surrender to the multi-master system bus are connected together.

The running arbiter would not pull the $\overline{\text{CBRQ}}$ line low—rather it is done by another arbiter seeking the services which pulls the $\overline{\text{CBRQ}}$ line low. The presently run arbiter then drops its $\overline{\text{BREQ}}$ signal and surrenders the bus, when proper surrender conditions exist. If $\overline{\text{CBRQ}}$ and ANYRQST are put into active conditions, the multi-master system bus would be surrendered after each transfer cycle.

12. Mention the methods of resolving priority amongst bus masters.

Ans. On a multi-master system bus, there may be several bus masters. The particular bus master which is going to gain control of multi-master system bus is determined by employing bus arbiters. Several techniques are there to resolve this priority amongst bus masters. They are:

- Parallel Priority Resolving Technique.
- Serial Priority Resolving Technique.
- Rotating Priority Resolving Technique.

13. Discuss the Parallel Priority Resolving Technique.

Ans. The technique of resolving priority in this scheme is shown in Fig. 19d.3. Four arbiters have been shown each of whose $\overline{\text{BREQ}}$ (Bus Request) output line is entered into a priority encoder and then to a decoder. The $\overline{\text{BPRN}}$ (Bus Priority In) output lines of the encoder are returned—one each to each of the arbiters.

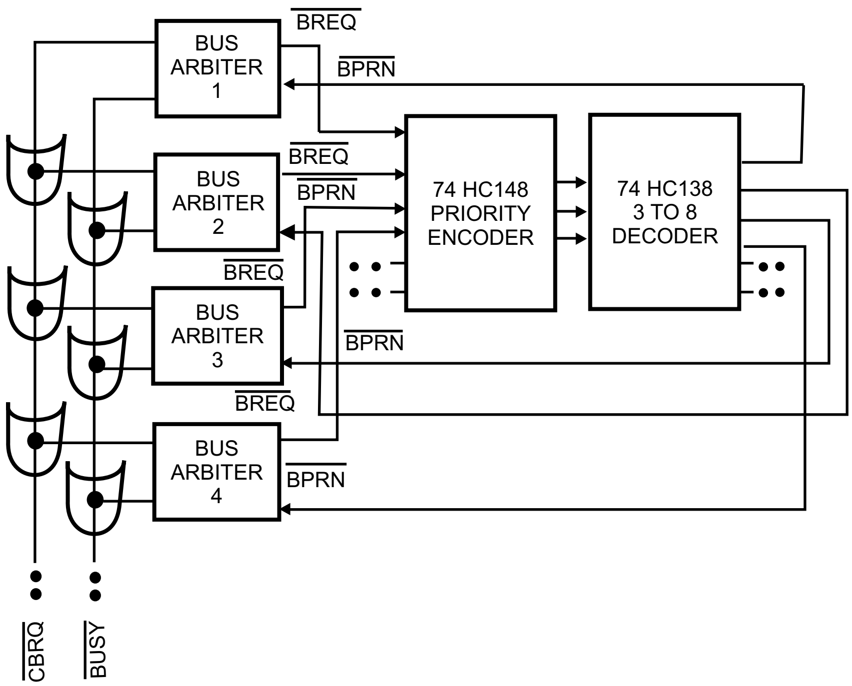


Fig. 19d.3: Parallel Priority Resolving Technique

But corresponding to the highest priority active $\overline{\text{BREQ}}$, an active $\overline{\text{BPRN}}$ is obtained, which activates the respective bus arbiter, neglecting the lower priority $\overline{\text{BREQ's}}$. Thus the bus master corresponding to this bus arbiter will identify itself with the multi-system bus master or would wait until the present bus transaction is complete.

Fig. 19d.4 shows the waveform timing diagrams of BREQ, $\overline{\text{BPRN}}$ and $\overline{\text{BUSY}}$ signals, which are synchronised with $\overline{\text{BLCK}}$. The explanation of the waveform timing diagram is as follows. When the bus cycles are running, the $\overline{\text{BREQ}}$ line goes low [①]. There can be more than one $\overline{\text{BREQ}}$ line going low during this time. But the 74HC138 3 to 8 decoder would output a low on that particular $\overline{\text{BPRN}}$ [②] which corresponds to the

highest priority $\overline{BREQ}$. The time between ② and ③ is the time for the presently run bus cycles to be completed at which time the active arbiter pulls its $\overline{BUSY}$ high,

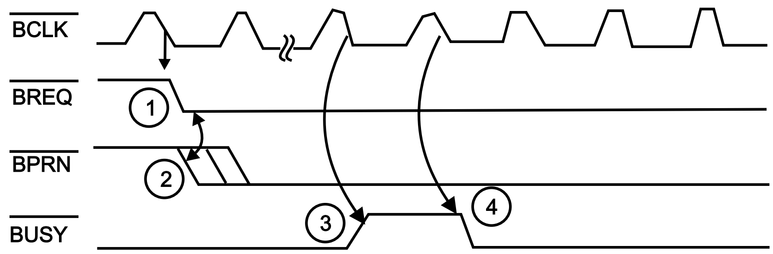


Fig. 19d.4: Higher Priority Arbiter obtaining the bus from A lower Priority Arbiter

thereby pulling it off from the multi-master system bus. In the next $\overline{BLCK}$ cycle, the arbiter which just had the right to use the system bus, pulls its own $\overline{BUSY}$ line low, thereby making it active and at the same time forcing other arbiters off the bus.

14. Discuss the Serial Priority Resolving Technique.

Ans. Figure 19d.5 shows the technique employed in this scheme. This scheme does away with the hardware combination of encoder-decoder logic as employed in Parallel Priority Scheme. Instead, the higher priority bus arbiter's $\overline{BPRO}$ (Bus Priority Out) is fed to the $\overline{BPRN}$ (Bus Priority In) of the just next lower priority one.

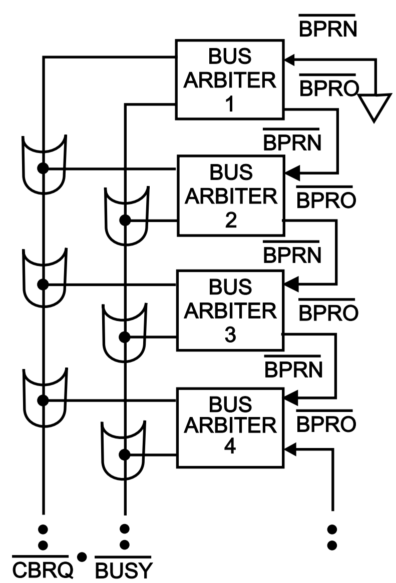


Fig: 19d.5: Serial Priority Resolving

15. Discuss Rotating Priority Resolving Technique.

Ans. In this scheme, the priority, to get the right to use the multi-master system bus, is dynamically reassigned. The circuitry is so designed that each of the requesting arbiters gets an equal chance to use the multi-master system bus.

16. Compare the three types of Priority Resolving Techniques.

Ans. In the serial priority scheme, the number of arbiters that may be daisy-chained together is a function of $\overline{\text{BLCK}}$, as well as the propagation delay that exists from one arbiter to the next one. With a 10 MHz frequency of operation, a maximum of 3 arbiters can be so connected.

The rotating priority resolving technique employs a considerable amount of external logic for its implementation. The parallel priority resolving technique is a good compromise compared to the other two in the sense that it employs a moderate amount of hardware to implement it while at the same time accommodating a good number of arbiters.

17. Discuss the modes of operations of 8289.

Ans. 8289 bus arbiter provides support to two types of processors : 8089 I/O Processor and 8086/8088. Thus 8289 supports two modes of operations—(a) IOB (I/O Peripheral Bus) mode —which permits the processor access to both I/O peripheral bus and a multi-master system bus. When 8289 needs to communicate with system memory, this is effected with the help of system memory bus. (b) RESB (Resident Bus) mode—which permits the processor to access to resident bus and a multi-master system bus.

All devices residing on IOB are treated as I/O devices (including memory) and are all addressed by I/O commands, the memory commands being handled by multimaster system bus. A Resident Bus can also issue both memory and I/O commands—but distinct from the multi-master system bus. The Resident Bus has only one master.

When $\overline{\text{IOB}}=0$, 8289 is in IOB mode and when RESB = 1, 8289 is in RESB mode. When $\overline{\text{IOB}}$ =0 and RESB = 1, then 8089 interfaces with 8086 to a multi-master system bus, a Resident Bus and an I/O Bus. Again, for $\overline{\text{IOB}}$ = 1 and RESB = 0, 8089 interfaces with 8086 to a multi-master system bus only.

Index